ANÁLISIS E IMPLEMENTACIÓN DEL ALGORTIMO GENÉTICO DE CHU-BEASLEY PARA RESOLVER EL PROBLEMA DEL AGENTE VIAJERO (TSP) Y SU VARIANTE, EL PROBLEMA DE RUTAS DE VEHÍCULO (VRP)

CLAUDIA PATRICIA ARIAS HERNÁNDEZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA-2015

ANÁLISIS E IMPLEMENTACIÓN DEL ALGORTIMO GENÉTICO DE CHU-BEASLEY PARA RESOLVER EL PROBLEMA DEL AGENTE VIAJERO (TSP) Y SU VARIANTE, EL PROBLEMA DE RUTAS DE VEHÍCULO (VRP)

CLAUDIA PATRICIA ARIAS HERNÁNDEZ

TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN

DIRECTOR

Ph.D (c) JORGE IVÁN RÍOS PATIÑO

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA-2015

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Pereira, Risaralda. Junio de 2015

A Dios y a la Virgen y a mi familia por el apoyo y el amor incondicional que me han brindado.

AGRADECIMIENTOS

Primero a Dios y a la Virgen, quienes guían y protegen a diario mi camino. A mi padre Luis Alfonso, quien siempre está con nosotras y somos la principal prioridad en su vida, por la gran valentía, coraje y tenacidad con la que lucha día a día por nuestro bienestar, porque siempre se ha esforzado para que nunca falte nada y por enseñarme con su ejemplo lo que es el esfuerzo, el trabajo y la responsabilidad. A mi madre Liliana Patricia por estar siempre conmigo, por el apoyo, el amor y los sabios consejos que me ha brindado, fuente de inspiración para mi desarrollo como persona, que con su demostración de una madre ejemplar me ha enseñado a no desfallecer ni rendirme ante nada y a siempre perseverar. A mi hermana María Camila por siempre creer en mí. Agradezco al Ph.D (c) Jorge Iván Ríos Patiño por su apoyo incondicional, confianza, orientación y colaboración en el desarrollo de este trabajo, a todos mis profesores, quienes me orientaron con profesionalismo ético en la adquisición de conocimientos, afianzado mi formación. A la Universidad Tecnológica de Pereira por brindarme la oportunidad de realizar mis estudios. A mis compañeros, quienes compartieron conmigo durante este tiempo y a todas aquellas personas que estuvieron presentes en mi desarrollo personal y profesional.

Muchas Gracias

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El problema del agente viajero (TSP) y el problema de rutas de vehículo (VRP) han sido problemas de gran importancia e influencia a lo largo de la historia, por tal razón las investigaciones y estudios se han enfocado en implementar diversos algoritmos que permitan encontrar una solución optima a estos problemas, buscando dar un aporte significativo en alguna de las áreas involucradas; debido a que estos problemas son considerados difíciles de resolver y dentro de la optimización combinatoria son conocidos como problemas NP-Hard, pues de estos no se obtiene una solución de manera eficiente; así mismo dentro de la teoría de la complejidad computacional pertenecen a la clase NP-Completos, lo que indica que no se puede garantizar encontrar la mejor solución en un tiempo de cómputo razonable, ya que este aumenta exponencialmente, generando así la búsqueda de soluciones aproximadas, para lo cual, es conveniente emplear métodos heurísticos y metaheurísticos que aplican el conocimiento del problema para acercarse a la solución de este en un tiempo de computo razonable.

En este trabajo se llevará a cabo la implementación del algoritmo genético de Chu-Beasley (AGCB), el cual presenta un proceso evolutivo altamente eficiente, que se desarrollará de forma clásica llevando a cabo los procesos de evolución natural como la *selección*, en el cual se determinan los padres aptos para pasar a la siguiente generación; la *recombinación*, en donde los cromosomas de los padres se comparten para generar nuevos individuos candidatos y así determinar cuál de los nuevos individuos sigue en el proceso, teniendo en cuenta que posea la mejor función objetivo; y finalmente la *mutación* en donde se controla la factibilidad nuevamente y se emplea una tasa que es ajustable dentro de los parámetros generales del algoritmo con el fin de alterar alguno de los alelos de forma aleatoria.

ABSTRACT

The traveling salesman problem (TSP) and vehicle routing problem (VRP) have been issues of great importance and influence throughout history, for that reason research and studies have focused on implementing various algorithms to find optimal solution to these problems, seeking to make a significant contribution in any of the areas involved; Because these problems are considered difficult to solve and within combinatorial optimization are known as NP-hard problems because these are not efficient solution is obtained; Likewise in the computational complexity theory they belong to the NP-complete class, indicating that you can not guarantee to find the best solution in a reasonable computation time, as this increases exponentially, generating finding approximate solutions, for which it is desirable to employ heuristic and metaheuristic methods that apply knowledge to approach problem solving is in reasonable computation time.

This work will be carried out the implementation of the genetic algorithm Chu-Beasley, which has a highly efficient evolutionary process, which will conventionally carrying out the processes of natural evolution and selection, in which unfit parents are determined to move to the next generation; recombination, where parental chromosomes shared candidates to generate new individuals and thus determine which of the new individuals still in the process, taking into account that has the best objective function; and finally mutation where feasibility is controlled again and a rate that is adjustable within the general parameters of the algorithm in order to alter any of the alleles randomly used.

GLOSARIO

ALGORITMO

Procedimiento computacional eficaz empleado para dar solución a un problema específico, el cual mediante una entrada o estado inicial, realiza la ejecución de una secuencia de pasos o instrucciones con el fin de obtener una salida.

ALGORITMO GENÉTICO

Surge en los años 1970, de la mano de John Henry Holland y se inspiran en la teoría de la evolución biológica de Darwin, es un método de búsqueda que permite encontrar la solución a problemas, partiendo desde una población inicial en la cual se seleccionan los individuos más aptos o mejor capacitados para después reproducirlos y mutarlos con el fin de obtener una nueva población de individuos que estarán mejor adaptados y poseerán mejores condiciones y características que la generación anterior.

ALGORITMO DE GENÉTICO CHU-BEASLEY (AGCB)

El algoritmo genético de Chu-Beasley es una versión modificada del algoritmo genético básico, se fundamenta en mantener durante todo el proceso la diversidad de los individuos que conforman la población, este permite que en cada generación solo un individuo sea reemplazado, siempre que cumpla con las condiciones de diversidad, optimalidad y/o factibilidad previamente establecidas. Su proceso evolutivo se genera de manera clásica llevando a cabo procesos como la selección, la recombinación, la mutación y la aceptación que implica que si el nuevo individuo es mejor que el individuo de peor calidad este se reemplaza directamente, de lo contrario se le permite que pase a la siguiente generación.

COMPLEJIDAD COMPUTACIONAL

Medición de los recursos computacionales (tales como, tiempo de ejecución y espacio de almacenamiento), que son empleados en la solución de un problema, con el fin de determinar la eficiencia de los algoritmos considerando la viabilidad de este en cuanto a la optimización de costo y de tiempo.

COMPUTACIÓN EVOLUTIVA

Es una rama de la inteligencia artificial que comprende técnicas o herramientas computacionales basadas en la teoría de la evolución biológica, que buscan soluciones para aquellos problemas de optimización combinatoria.

CRUCE

Operador genético que simula el proceso de intercambio de segmentos de material genético entre dos cromosomas, conocido como la *reproducción sexual.* Este se realiza mediante la elección de una pareja de individuos para combinar sus características genéticas y producir una descendencia con mejores aptitudes.

ESTRATEGIAS EVOLUTIVAS

Técnicas computacionales basadas en los procesos evolutivos biológicos tales como la selección, la mutación y la recombinación, que son empleadas en la solución de problemas de optimización.

HEURÍSTICAS

Los métodos heurísticos permiten resolver problemas de optimización, los cuales son procedimientos eficientes que garantizan encontrar soluciones de alta calidad con un costo computacional razonable, aunque estas soluciones no son exactas, establece un grado de optimalidad o factibilidad mayor.

METAHEURÍSTICAS

Las metaheurísticas son empleadas en la búsqueda de la solución de aquellos problemas que generalmente no tienen una heurística o un algoritmo que les permita encontrar una solución adecuada, son diseñadas especialmente para aquellos problemas difíciles de resolver y que hacen parte de la optimización combinatoria. Las soluciones que se obtienen de estas son mejores que las que dan las heurísticas clásicas, aunque requieren de un tiempo de cómputo mayor. Las metaheurísticas son considerados algoritmos híbridos que permiten combinar diferentes métodos y mecanismos relacionados con la inteligencia artificial y la evolución biológica.

MUTACIÓN

Operador genético que permite la preservación de la diversidad genética en una población, mediante la alteración o modificación aleatoria de los genes, evitando la convergencia prematura en la búsqueda de soluciones.

OPERADORES GENÉTICOS

Procedimientos llevados a cabo dentro de los algoritmos genéticos que permiten mantener la diversidad dentro de una población, en donde estos procesos tiene similitud con los procesos llevados en el proceso evolutivo biológico. Dentro de los operadores genéticos que más se utilizan son la mutación, recombinación y cruce.

PROBLEMA DEL AGENTE VIAJERO

Es uno de los problemas más famosos y estudiados, debido a la complejidad de su solución y a la sencillez de su planteamiento, sin embargo ningún método de solución eficaz es conocido para el caso general de este. El problema del agente viajero (TSP) consiste en que dadas n ciudades y la distancia entre ellas, se

deberá encontrar la ruta más corta posible de ir desde el punto de partida y visitar cada ciudad una sola vez retornando a la ciudad de origen.

PROBLEMA DE RUTEO DE VEHÍCULO (VRP)

El problema de rutas de vehículo, es un problema de optimización en el cual se deben establecer un conjunto de rutas para una flota de vehículos que parten de uno o más depósitos con el fin de satisfacer la demanda de sus clientes ubicados geográficamente a diferentes distancias minimizando el costo que implica realizar esta operación.

PROGRAMACIÓN EVOLUTIVA

La programación evolutiva fue propuesta en la década de los 60’s por Lawrence J. Fogel, la cual hace parte de la computación evolutiva. Esta es una variación de los algoritmos genéticos en la cual es considera la inteligencia como un comportamiento de adaptación de cada uno de los individuos de una población.

SELECCIÓN

Operador genético que basado en la teoría de Darwin, elije en una población aquellos individuos más aptos, con capacidad de adaptación y probabilidad de reproducción mayor.

INTRODUCCIÓN

Las investigaciones sobre la complejidad de los algoritmos se han llevado a cabo desde hace varios años, antes de que se realizaran las investigaciones sobre este tema, se evidenciaron diferentes estudios que sirvieron de base para su desarrollo. En el año 1844 Gabriel Lamé realizó un ejemplo de análisis de complejidad conocido como el “algoritmo de Euclides”, el cual fue uno de los primeros estudios realizados que tuvo un enfoque de análisis de tiempo de ejecución, más adelante en el año 1936 se dio uno de los aportes de mayor influencia conocido como la Máquina de Turing y aunque en la década de los 40’s y 50’s esta demostraba ser un modelo teórico correcto de computo, se descubrió que su modelo básico no lograba cuantificar el tiempo y la memoria requerida por una computadora, problema que en la actualidad y a lo largo de la historia se ha tratado y ha sido objeto de estudio de muchos investigadores en el mundo.

Una vez es evidenciado este problema a principios de los 60’s Hartmanis and Stearns plantearon la idea de medir el tiempo y espacio como una función de la longitud de entrada, generando así, el surgimiento de la teoría de la complejidad computacional que llevó a que varios investigadores guiaran sus estudios en entender y conocer las medidas de complejidad y la relación que tenían entre ellas, es así como en 1965 Edmonds, define que un buen algoritmo es aquel cuyo tiempo de ejecución es polinómico, lo cual condujo al surgimiento del concepto de los problemas NP-Completos; problemas que el investigador norteamericano Stephen Cook y el investigador soviético Leonid Levin, de manera independiente, probaron su existencia y que más adelante en 1972, Richard Karp demostró que 21 problemas combinatorios y de teoría de grafos, los cuales son caracterizados por ser intratables computacionalmente, pertenecían a la clase de los problemas NP-Completos. En la década de los 80’s y 90’s las investigaciones sobre las clases de complejidad y las técnicas combinatorias para entender estos modelos

aumentaron y en la actualidad se siguen realizando estudios y empleando procedimientos con el fin de dar solución a estos problemas.

Dentro de los problemas de complejidad computacional más famosos y estudiados a lo largo de la historia se encuentran, el problema del agente viajero (TSP) y el problema de rutas de vehículo (VRP), los cuales han sido problemas de gran importancia e influencia en la investigación en las diferentes áreas tales como la inteligencia artificial, las matemáticas, la física, la biología, la investigación de operaciones, el transporte, la genética, entre otras; es por esto que se han implementado diversos algoritmos que permitan encontrar una solución optima a estos problemas, buscando dar un aporte significativo en alguna de las áreas involucradas; debido a que estos problemas son considerados difíciles de resolver y dentro de la optimización combinatoria pertenecen a la clase NP-Hard, pues de estos no se obtiene una solución de manera eficiente; así mismo dentro de la teoría de la complejidad computacional pertenecen a la clase NP-Completos, lo que indica que no se puede garantizar encontrar la mejor solución en un tiempo de cómputo razonable, ya que este aumenta exponencialmente o en muchos casos factorialmente, generando así la búsqueda de soluciones aproximadas, para lo cual, es conveniente emplear métodos heurísticos o métodos metaheurísticos que aplican el conocimiento del problema para acercarse a la solución de este en un tiempo razonable.

Estos problemas se pueden considerar desde dos puntos vista, el práctico y el teórico, en el primero estos no están resueltos y en el segundo las técnicas que se han ido empleando son solo aproximaciones suficientemente aceptables pero no son una solución real de estos problemas. Buscando estas posibles soluciones se han empleado los algoritmos genéticos, que simulando la evolución biológica, han basado su configuración en determinados aspectos e incluyendo distintas variaciones con el fin de encontrar una solución con un nivel de eficiencia y optimización mayor.

Los algoritmos genéticos a través de los años han mostrado que tienen la capacidad de encontrar soluciones óptimas para los problemas del agente viajero (TSP) y el problema de rutas de vehículo (VRP), muchas de las investigaciones que se conocen en la actualidad han propuesto diversas formas de implementarlos, con el fin de encontrar la mejor solución o la óptima respuesta, por esta razón las implementaciones han ido más allá y muchos de estos estudios se han dirigido hacia el desarrollo de algoritmos híbridos que en conjunto con otras metodologías permitan que la búsqueda de estas soluciones sea más eficiente.

En este trabajo se desarrolla la implementación del algoritmo genético de Chu-Beasley (AGCB), el cual presenta un proceso evolutivo altamente eficiente, que siendo un algoritmo genético básico posee la característica fundamental de mantener durante todo el proceso la diversidad de los individuos presentes en la población, reemplazando en cada generación un solo individuo, siempre que este cumpla con las condiciones establecidas de optimalidad y factibilidad.

La implementación del algoritmo genético de Chu-Beasley (AGCB) para la solución de los problemas del agente viajero (TSP) y el problema de rutas de vehículo (VRP) se desarrollará de forma clásica llevando a cabo los procesos de *selección*, en el cual se determinan los padres aptos para pasar a la siguiente generación; de *recombinación*, en donde los cromosomas de los padres se comparten para generar nuevos individuos candidatos y así determinar cuál de los nuevos individuos sigue en el proceso, teniendo en cuenta que posea la mejor función objetivo; y finalmente la *mutación* en donde se controla la factibilidad nuevamente y se emplea una tasa que es ajustable dentro de los parámetros generales del algoritmo con el fin de alterar alguno de los alelos de forma aleatoria.

1. ALGORITMOS GENÉTICOS

1.1 INTRODUCCIÓN

En la búsqueda de métodos que dieran solución a aquellos problemas clasificados como difíciles de resolver, los cuales abordan temas tanto de búsqueda como de optimización, se plantearon los algoritmos genéticos, los cuales mediante su planteamiento basado en la evolución y el comportamiento natural generan soluciones aproximadas o soluciones optimas para problemas del mundo real.

Los algoritmos genéticos realizan la simulación de los procesos llevados a cabo por los organismos vivos, empleando métodos evolutivos tales como la selección, la herencia, la mutación y la recombinación, con el fin de encontrar individuos que posean las mejores características y aptitudes, generando así una evolución significativa para las especies, por esta razón los algoritmos genéticos han sido empleados en la solución de aquellos problemas en los cuales no es posible encontrar una respuesta exacta, dado que permiten que una vez sean elegidos los individuos con mejor grado de adaptación se garantice que la nueva población generada converja hacia una solución óptima del problema.

Como se mencionó anteriormente los algoritmos genéticos son métodos que aunque no garantizan la solución exacta de los problemas, si proporcionan una solución cercana a la óptima en un tiempo aceptable, por esta razón los algoritmos genéticos se emplean en la solución de problemas famosos y que han sido objeto de estudio a lo largo de la historia, tales como el problema del agente viajero (TSP) y el problema de ruteo de vehículos (VRP).

1.2 HISTORIA

En los años 1950 y 1960 se iniciaron los estudios e investigaciones sobre el empleo de los sistemas evolutivos como métodos o herramientas para tratar problemas de optimización. En el año 1957 Alex Fraser realizó la publicación de diferentes artículos sobre *selección natural,* lo cual permitió el inicio de la simulación por medio de los computadores de la evolución biológica, en las cuales se incluyeron elementos principales de los algoritmos genéticos.

En la década de 1960 Rechenberg planteó un método para optimizar parámetros empleados por ciertos dispositivos, lo cual se dio a conocer como *"estrategias evolutivas",* tiempo después, en el año 1966 Fogel, Owens y Walsh, crearon la "programación evolutiva" la cual empleaba una máquina de estados en donde los diagramas de estados de transición evolucionaban mediante mutación aleatoria para así encontrar la mejor solución. En el año 1967 Bagley mencionó por primera vez el término de Algoritmos genéticos, el cual empleó para la búsqueda de conjuntos de parámetros en funciones de evaluación de juegos, aunque es a John Holland a quien se considera el creador de los algoritmos genéticos, gracias a su libro "*Adaptación en Sistemas Naturales y Artificiales" (1975),* en el cual describió a estos algoritmos como una abstracción de la evolución biológica, en donde su método consistía en desplazarse de una población a una nueva, empleando un sistema de selección similar a la *"selección natural"*, así mismo utilizando operadores genéticos como cruces, mutaciones e inversión.

El crecimiento de la investigación sobre los algoritmos genéticos a partir del año 1990 se vio reflejado en la cantidad de libros, publicaciones y conferencias que se dieron con base a este tema, además de las técnicas que ha aportado la Computación Evolutiva permitiendo el intercambio de ideas y la aparición de nuevas técnicas hibridas que buscan generar soluciones más optimas.

En la actualidad se siguen usando los algoritmos genéticos y desarrollando aplicaciones basadas en estos ellos, abarcando áreas como la ingeniería, la investigación de operaciones, la programación automática, entre otras; debido a que pueden tratar con éxito muchos de los problemas para los cuales no existen técnicas o métodos especializados para resolverlos e incluso, para aquellos que si las poseen pueden generar muy buenas soluciones.

1.3 DEFINICIÓN

Los algoritmos genéticos son un método de búsqueda heurística que imita la teoría de la evolución biológica de Darwin (1859)[1] generando soluciones a problemas relacionados con optimización y búsqueda, empleando operadores genéticos tales como selección, mutación, herencia y recombinación.

John Holland y sus estudiantes de la Universidad de Michigan basaron sus estudios en la búsqueda de la explicación del proceso adaptativo de los sistemas naturales y en el diseño de sistemas que de manera artificial simularan los procesos o mecanismos más importantes realizados por estos sistemas. Con base a lo anterior Goldberg (1989) definió los algoritmos genéticos como "*mecanismos de selección natural y genética natural. Combinan la supervivencia de los más compatibles entre las estructuras de cadenas, con una estructura de información ya aleatorizada, intercambiada para construir un algoritmo de búsqueda con algunas de las capacidades de innovación de la búsqueda humana"*[2].

El funcionamiento de los algoritmos genéticos parte de una población inicial o de una muestra aleatoria elegida en el espacio de búsqueda, en donde se seleccionan los individuos que tengan la mayor capacidad de adaptación, las

[1] Darwin,Charles (1859), *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life* (1.ª Edición), Londres: John Murray

[2] Goldberg, D. (1989) Genetics Algorithms in Search, Optimization and Machine Learning. Addison Wesley.

mejores aptitudes y características, para después empleando los operadores genéticos reproducirlos, mutarlos y recombinarlos, con el fin de garantizar que la nueva generación posea individuos mejor adaptados que la anterior. En la figura 1 se muestra el funcionamiento de los algoritmo genético.

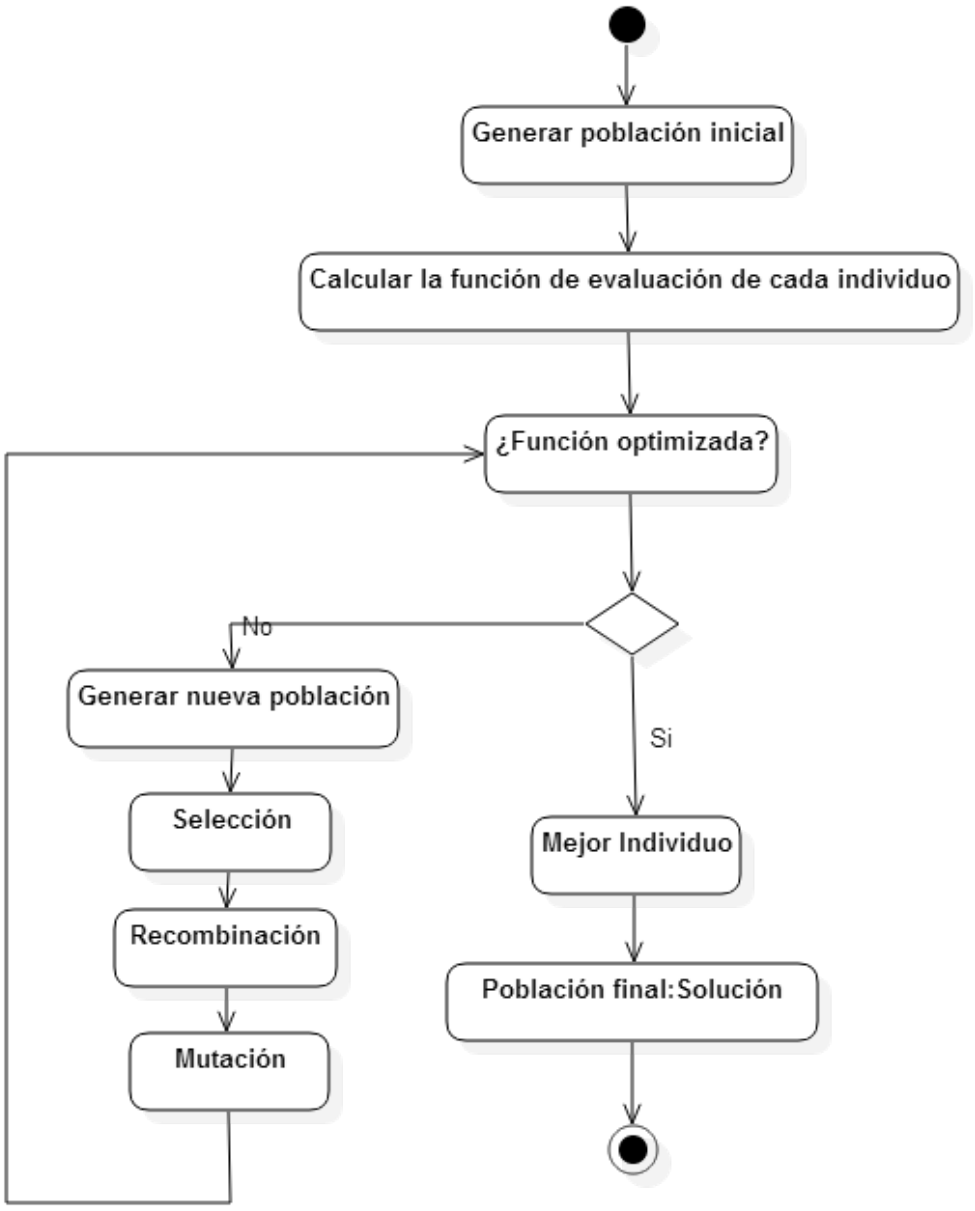

Figura 1. Funcionamiento de los algoritmos genéticos.

1.4 CODIFICACIÓN DE LA POBLACIÓN INICIAL

Los algoritmos genéticos están compuestos por individuos, los cuales son las posibles soluciones del problema y se representan como un conjunto de parámetros denominados *genes,* estos de manera conjunta forman una cadena de valores (en este caso una cadena de bits), la cual es conocida como *cromosoma*, cada gen representa una posición en la cadena. La representación de los genes se puede realizar por una serie de símbolos que pertenezcan a un determinado alfabeto, por lo cual pueden ser representados a través de un sistema de tipo numérico, en donde cada uno de estos valores, que son asignados a los genes, son denominados *alelos.*

La codificación que se ha empleado desde los inicios de los estudios e investigaciones planteadas por John Holland, ha sido la codificación binaria, en donde solo existirán dos valores para los alelos por cada gen, que son el cero y el uno {0,1}.

El método de codificación de cada uno de los puntos del espacio de búsqueda con los que se va a trabajar es la característica más importante del algoritmo, dado que una vez se aplique la codificación, el dominio de la función a optimizar se transforma en el conjunto de valores representados por cadenas acotadas e independientes de su origen. La figura 2 muestra la representación de cada uno de los conceptos mencionados anteriormente.

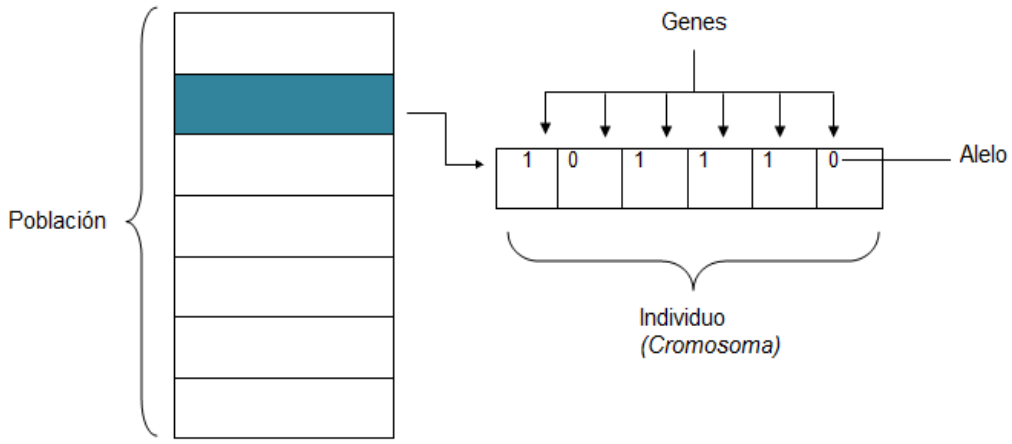

Figura 2. Componentes de un algoritmo genético

1.5 OPERADORES GÉNETICOS

Dentro de los cromosomas existen una serie de variables que los algoritmos genéticos aplicaran en cada uno de ellos, con la finalidad de encontrar la mejor solución en un ambiente en el cual se encuentran otras soluciones, que permitirán que la presión selectiva sobre la población genere que solo sobrevivan los individuos mejor adaptados y puedan transmitir su material genético a las siguientes generaciones, lo cual se realizará mediante las mutaciones y cruces.

Los algoritmos genéticos emplean tres operadores genéticos que han sido considerados fundamentales a lo largo de los estudios e investigaciones llevadas a cabo, los cuales son la *selección*, *cruce* y la *mutación*; estos operadores son de gran importancia en el funcionamiento del algoritmo debido a que se encargan de dirigir la población inicial a través de una serie de procedimientos que al convertirla

en nuevas generaciones, con individuos mucho más adaptados y con mejores características, generan soluciones que convergen a un óptimo global.

1.5.1 OPERADOR GENÉTICO DE SELECCIÓN

En la fase de reproducción o de selección se eligen los individuos de la población para cruzarlos y mutarlos produciendo una nueva generación con los individuos que posean las mejores aptitudes y características, que de acuerdo con los planteamientos de la teoría de Darwin en esta nueva descendencia sobrevivirán únicamente los individuos con mayor capacidad de adaptación.

La selección de los individuos para la creación de la nueva generación, es fundamental y decisiva en el comportamiento del algoritmo genético, es por esta razón que existen diferentes métodos de selección, dentro de los que se destacan los siguientes:

Selección por Rueda de Ruleta

Este tipo de selección fue propuesto por DeJong, el cual emplea un círculo o ruleta que se encuentra divido en sectores circulares que son proporcionales a la función objetivo. Cada uno de los cromosomas o individuos tienen una parte de la ruleta y según su nivel de adaptación esta puede ser mayor o menor.

Selección por torneo

Este método consiste en realizar una selección con base a comparaciones directas entre un pequeño subconjunto de individuos elegidos al azar desde la población. En esta selección se elige el individuo, que al realizar la comparación, tenga la mayor puntuación para después ser reproducido. La selección por torneo

no requiere comprar la totalidad de la población lo que genera convergencia prematura.

Selección por Ranqueo

Este tipo de selección asigna a cada individuo o cromosoma un rango numérico basado en sus características, esta selección presenta convergencia prematura, la cual puede ser tratada pero significaría un costo computacional, debido a que se debe ordenar la población inicial.

Otros de los métodos de selección que se emplean son:

- Selección Jerárquica
- Selección por estado estacionario
- Selección escalada
- Selección elitista

1.5.2 OPERADOR GENÉTICO DE CRUCE

El operador de cruce es uno de los operadores principales dentro de los algoritmos genéticos, ya que permite que se generen nuevos individuos o cromosomas tomando las características de su progenitor o padre.

Dentro de los tipos de operadores de cruce existentes son:

Cruce simple o cruce de un solo punto

Este cruce consiste en elegir un punto aleatorio sobre dos individuos padres y después realizar la división de cada padre en el punto de cruce elegido. La información genética de uno de los padres desde la parte inicial hasta el punto de cruce se copia y el resto de información es copiada de la parte restante del otro padre. Un ejemplo de esto se muestra en la figura 3.

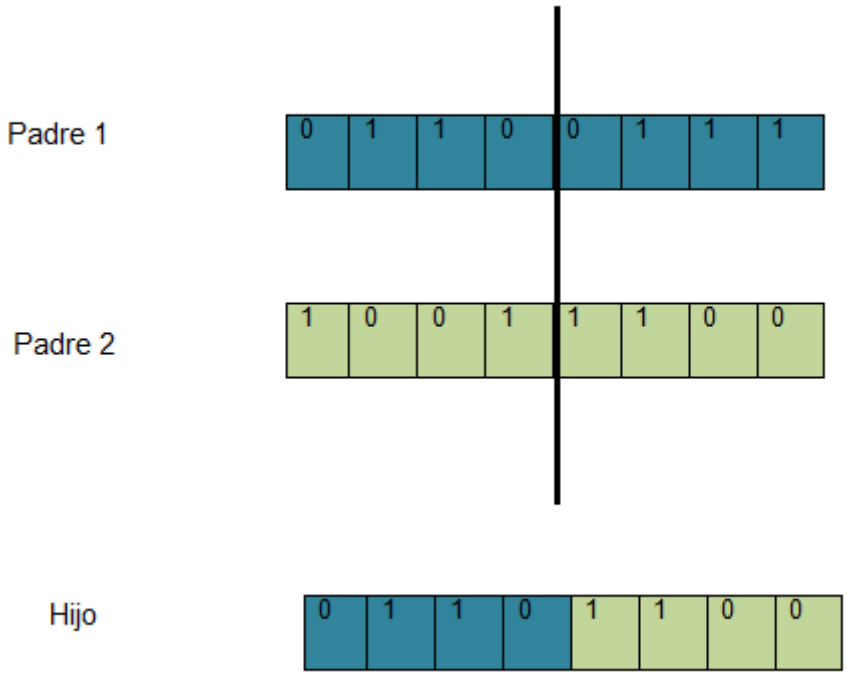

Figura 3. Cruce de un solo punto.

Cruce de dos puntos

El cruce de dos puntos es similar al anterior, con la diferencia de que se seleccionan aleatoriamente dos puntos de cruce, en donde se copiará al hijo los genes de su padre hasta el primer punto de cruce, y los genes del otro padre hasta el segundo y de este en adelante se copia la información genética del otro padre hasta el final. En la figura 4 se muestra como se lleva a cabo este tipo de cruce:

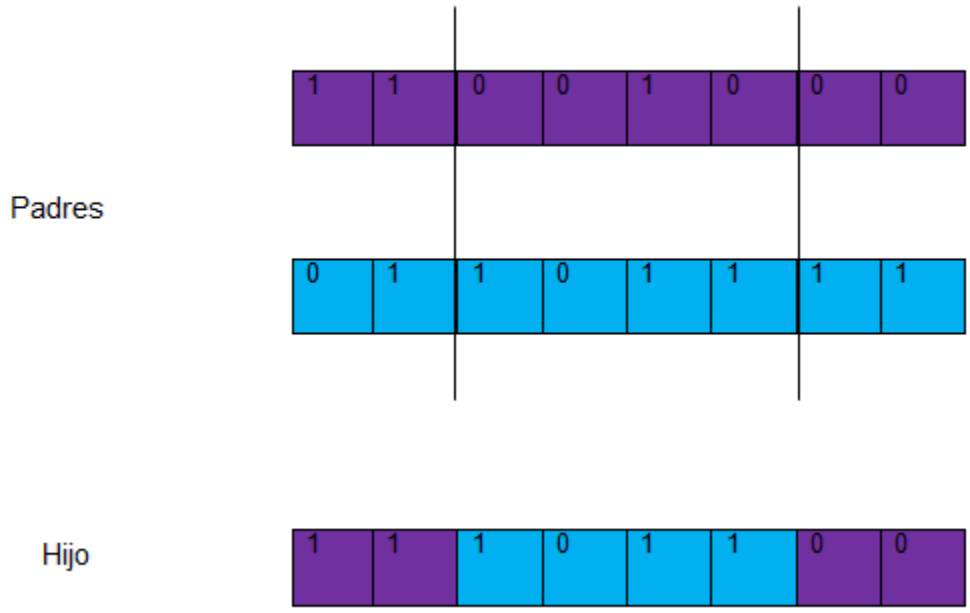

Figura 4. Cruce de dos puntos

Cruce uniforme

En el cruce uniforme existe la posibilidad de que cada gen del individuo descendiente pertenezca a alguno de los dos padres.

1.5.3 OPERADOR GENÉTICO DE MUTACIÓN

El operador de mutación es de gran beneficio, ya que contribuye a que la diversificación genética se dé en una población, impidiendo que las soluciones generadas se vean limitadas por óptimos locales. La mutación modifica ciertos genes de forma aleatoria, la cual depende de la codificación del problema y de la selección.

El proceso de mutación se puede realizar de diversas formas, dos de las cuales son:

- *Inversión de genes*, en donde se seleccionan los genes de manera aleatoria y se invierte su valor, en el caso de codificación numérica se sustituye un número por otro, por ejemplo se cambia un cero 0 por un 1 o viceversa.

- *Cambio de orden*, en el cual se seleccionan dos genes de manera aleatoria y se realiza el intercambio de sus posiciones.

En la figura 5 se muestra el proceso de mutación con inversión de genes.

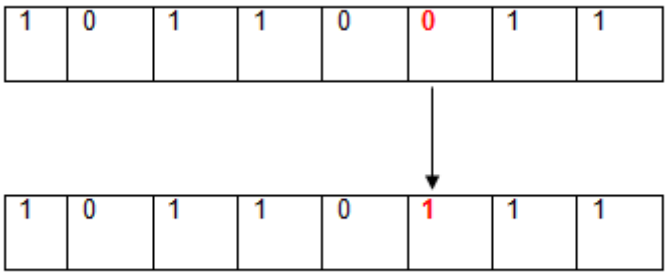

Figura 5. Mutación con inversión de genes.

2. PROBLEMA DEL AGENTE VIAJERO (TSP)

2.1 INTRODUCCIÓN

El problema del agente viajero o también conocido como TSP (en inglés, Traveling Salesman Problem), es uno de los problemas más importantes y estudiados a lo largo de la historia debido a su complejidad computacional. Es clasificado dentro de los problemas NP-Hard, ya que no se ha logrado construir, hasta el momento, algoritmos que encuentren la solución exacta en tiempo polinomial en cualquier instancia de este, a menos que se demuestre que los problemas P son iguales a los NP.

El origen del TSP se dio en la década de 1930 el cual fue estudiado por primera vez en Viena y Harvard, especialmente por el economista y matemático Karl Menger, poco tiempo después en la Universidad de Princeton se introdujo el nombre de “problema del agente viajero” por Hassler Whitney; la importancia de este problema fue aumentando y cada vez se hizo más reconocido debido a su planteamiento simple y a la cantidad de aplicaciones que posee en diferentes áreas, tales como la industria, la logística, el transporte, la genética, entre otras. El

TSP es un problema de optimización, el cual se genera cuando se tienen algunos clientes que demandan un determinado servicio y se debe encontrar la ruta óptima que permita satisfacer sus necesidades. La importancia del problema del agente viajero no solo comprende las diversas aplicaciones que posee, sino que la investigación y estudios realizados sobre este es aplicable y genera una base importante para la comprensión y el planteamiento de otros problemas de rutas que existen.

Encontrar soluciones óptimas y factibles para el problema del agente viajero significaría un gran beneficio en la optimización de recursos, es por esta razón que muchas de las investigaciones se han enfocado en implementar procedimientos que garanticen encontrar la mejor solución en un tiempo razonable. El problema del agente viajero ha sido empleado en el desarrollo de las nuevas técnicas que buscan dar solución a los problemas de optimización combinatoria, esto se debe a que posee las características adecuadas y habituales de estos problemas, con el fin de verificar la efectividad de los métodos y algoritmos, dado que su planteamiento es fácil de ilustrar y su solución resulta un desafío para los investigadores debido a su complejidad.

Para la solución del problema del agente viajero se han empleado técnicas que son solo aproximaciones suficientemente aceptables, pero no son una solución real y exacta, dentro de las técnicas se encuentran los métodos heurísticos o aproximados, los cuales generan soluciones razonables, aunque no se pueda evidenciar que sean las soluciones optimas, pero permite que mediante la utilización adecuada de la estructura del problema se pueda obtener una buena solución; otras de los métodos son los metaheurísticos los cuales son híbridos entre heurísticas y van más allá de las soluciones que pueda generar un método heurístico sin combinar. Los algoritmos genéticos son otras de las técnicas empleadas para la solución del problema del agente viajero, estos mediante la simulación de la selección natural y la adaptación evidenciada en la naturaleza y la implementación de los operadores genéticos (selección, recombinación y

mutación) generan nuevas poblaciones de posibles soluciones lo que favorece el proceso y permite que la búsqueda de la solución optima se genere de manera más precisa, aunque no exacta.

En esta sección se abarcarán los aspectos más importantes del problema del agente viajero, tales como el origen, la definición, la formulación del problema y demás aspectos que son necesarios conocer para lograr una adecuada interpretación del planteamiento del problema.

2.2 HISTORIA

El origen del problema del agente viajero no es posible determinarlo con precisión, pero a lo largo de la historia se han evidenciado estudios e investigaciones que han sido fundamentales en el planteamiento y definición de este problema**.** Hasta antes del año de 1800 el problema del agente viajero no había sido formulado matemáticamente, ya en este año el matemático irlandés Sir William Rowan Hamilton y el matemático británico Thomas Penyngton Kirkamn, realizaron la formulación matemática del problema y se dio a conocer el juego *Icosian de Hamilton* el cual consistía en que los jugadores debían completar recorridos por 20 puntos dados, usando únicamente las conexiones que se especificaban.

En el año 1925 H.M. Cleveland trabajó para la *Page Seed Company,* en donde se tenía una lista de ciudades en cinco hojas en las cuales se describía el Tour de Maine, en este primer esquema del tour se evidenció la eficiencia del recorrido, pero este no fue el único destino que Mr. Cleveland efectúo, dado que también realizó recorridos a través de Connecticut, Massachusetts, New York y Vermont, haciendo más de 1.000 paradas en total[3].

[3] Cook, J. William (2012), In Pursuit of the Traveling Salesman: *Mathematics at the Limits of Computation*,

La comunidad científica ha estado de acuerdo en que el término del “problema del agente viajero” se dio a conocer en la Universidad de Princeton entre los años 1931 y 1932. En la década de 1930, en la Universidad de Harward, Merrill Flood tuvo un papel de divulgación importante de este problema, ya que fue quien empezó a trabajar en la búsqueda de una ruta óptima para un autobús escolar; así mismo en Viena, el matemático Karl Menger enunció lo que se denominada el problema del mensajero planteando muchas de las propiedades del problema del agente viajero (TSP). En la década de 1940, fue estudiado por los estadísticos Mahalanobis (1940), Jessen (1942), Gosh (1948), Marks (1948).

En las décadas de los 50 y de los 60 el problema se popularizó y se iniciaron estudios con un número de ciudades mayor; G. Dantzig, R. Fulkerson y S. Johnson[4] en el año 1954 realizan una solución al problema del agente viajero, el cual es considerado uno de los principales eventos en la historia de la optimización combinatoria, en el cual se resuelve el problema de agente viajero para 49 ciudades, una por cada estado de EEUU. De ahí en adelante hasta la actualidad se han desarrollado diferentes algoritmos que puedan aplicarse a problemas con un número de ciudades cada vez más grande, gracias al gran desarrollo de la informática y a los estudios que se han venido realizando se han logrado avances importantes en la búsqueda de la mejor solución o la solución optima para este tipo de problemas.

En el siguiente cuadro se puede observar los problemas que fueron resueltos desde el año 1954 hasta el 2004:

New Jersey: Princeton University Press.

[4] G. Dantzig, R. Fulkerson y S. Johnson. Solution of large-scale traveling salesman problem*. The Rand Corporation, Santa Mónica, Californi*a. Agosto, 1954.

Tabla 1: Hitos en la resolución del problema del agente viajero (TSP)

Año	Equipo de Investigación	Tamaño de Instancia	Nombre
1954	G. Dantzig, R. Fulkerson, S. Johnson	49 ciudades	dantzig42
1971	M. Held y Karp RM	64 ciudades	64 puntos al azar
1975	PM Camerini, L. Fratta, F. Maffioli	67 ciudades	67 puntos aleatorios
1977	M. Grötschel	120 ciudades	GR120
1980	H. Crowder, MW Padberg	318 ciudades	lin318
1987	M. Padberg, G. Rinaldi	532 ciudades	att532
1987	M. Grötschel, O. Holanda	666 ciudades	gr666
1987	M. Padberg, G. Rinaldi	2.392 ciudades	pr2392
1994	D. Applegate, R. Bixby, V. Chvátal, W. Cook	7.397 ciudades	pla7397
1998	D. Applegate, R. Bixby, V. Chvátal, W. Cook	13.509 ciudades	usa13509
2001	D. Applegate, R. Bixby, V. Chvátal, W. Cook	15.112 ciudades	d15112

2004	D. Applegate, R. Bixby, V. Chvátal, W. Cook, K. Helsgaun	24.978 ciudades	sw24798

Fuente: Aida Calviño Martínez (2011). *Cooperación en los problemas del viajante (TSP) y de rutas de vehículos (VRP): una panorámica*

2.3 DEFINICIÓN

El problema del agente viajero o en inglés *Traveling Salesman Problem (TSP)*, es uno de los problemas más famosos y estudiados a lo largo de la historia en la investigación operativa, debido a la complejidad de su solución y a la sencillez de su planteamiento, se clasifica dentro de los problemas de optimización combinatoria NP-duros, además fue de los primeros problemas en estudiarse cuando surgió la teoría de la Complejidad Algorítmica, sin embargo ningún método de solución eficaz es conocido para el caso general de este. El problema del agente viajero (TSP) consiste en que dadas n ciudades y la distancia entre ellas, se debe encontrar la ruta más corta posible de ir desde el punto de partida y visitar cada ciudad una sola vez retornando a la ciudad de origen.

El problema del agente viajero se puede representar por medio de un grafo, en donde los nodos de este, representan cada una de las ciudades y los arcos la distancia existente entre un par de ellas, formando así lo que se conoce como el *ciclo o circuito hamiltoniano*.

Una de las definiciones apropiadas citadas en un reciente trabajo (“Problema del Agente Viajero, Métodos exactos de resolución”, 2014, p.14), para describir el ciclo hamiltoniano es la siguiente, *“Sea un grafo G se dice que un circuito o ciclo*

hamiltoniano es un ciclo simple que contiene a todos los vértices de G. Equivalentemente es una trayectoria que empieza y termina en el mismo vértice, no tiene aristas repetidas y pasa por cada vértice una única vez." Un ejemplo de la representación de un ciclo hamiltoniano se muestra en la siguiente imagen:

Figura 6. Ejemplo de circuito Hamiltoniano a través de África

Fuente: Cook, J. William (2012), *In Pursuit of the Traveling Salesman:Mathematics at the Limits of Computation*, (Ed.) New Jersey: Princeton University Press.

El problema del agente viajero, cuenta con diversas variantes[5] que aunque mantienen el mismo objetivo, han surgió cambios a través de las investigaciones y los estudios realizados, algunas de las variaciones más conocidas se describirán brevemente a continuación:

Problema del agente viajero generalizado (en inglés, Generalized Traveling Salesman Problem, GTSP)

El GTSP divide las ciudades que se deben recorrer en regiones, en donde se debe visitar una sola ciudad por región, buscando que se minimice el costo en el recorrido.

Problema de Ciclo Simple (en inglés, Simple Cycle Problem, SCP)

En esta variación del problema original, se tiene que existen costos para realizar las visitas de las ciudades y hay beneficios por cada una de estas, teniendo en cuenta que no tienen que estar incluidas todas las ciudades en el recorrido.

Problema del agente viajero con recogida y entrega de mercancías (en inglés, Pick-up and Delivery Traveling Salesman Problem, PDTSP)

En este problema se realiza el recorrido por las ciudades con un mismo vehículo, el cual posee una capacidad limitada para el transporte de productos, ya que debe proveer o recibir una cantidad determinada de estos en cada una de las ciudades incluidas en el recorrido.

Problema del agente viajero con ventanas de tiempo, (en inglés, Traveling Salesman Problem with Time Windows)

[5] García Travieso María Victoria (2014), *Problema del viajante de comercio (TSP), Métodos exactos de resolución. Universidad de La Laguna.*

Este problema posee la característica de manejo del tiempo, en donde para cada ciudad se tiene un tiempo establecido, el cual es mínimo para la llegada y máximo para la salida.

Problema del agente viajero con múltiples agentes

Esta variación establece la existencia de un número determinado de agentes, los cuales deben visitar las ciudades, con la limitante de que no podrán visitar aquellas ciudades que ya han sido visitadas por otro de los agentes que hacen parte de la ruta.

2.4 FORMULACIÓN DEL PROBLEMA DEL AGENTE VIAJERO (TSP)

El problema del agente viajero (TSP) se puede plantear matemáticamente, como el problema de hallar un ciclo hamiltoniano en un grafo completo en el cual $G = (V, E, C)$, donde se tiene que:

$V = \{1, 2, 3, \ldots, n\}$ es el conjunto de nodos o vértices que representan cada una de las ciudades del problema.

E, es el conjunto de ramas o aristas.

C, es la función de costo o distancia, la cual está dada por $C = (c_{ij})$, en donde cada $(i,j) \in E$ y se asigna C_{ij} para asignar el costo o la distancia entre las ciudades *i* y *j.*

En el desarrollo del problema del agente viajero también se tienen en cuenta los siguientes conceptos que hacen parte de la estructura del problema buscando que se cumpla el objetivo de hacer un recorrido a través de *n* ciudades, visitando a cada ciudad en una única ocasión:

- Un *camino* será la sucesión de aristas (a_1, a_2, a_3,…., a_k) en donde se tiene que el vértice final de cada una de las aristas coincide con el inicial de la siguiente.
- Un ciclo es defino como el camino *(a_1, a_2, a_3,…., a_k)* en el cual vértice a_k, que es el vértice final, coincide con el vértice inicial a_1. De acuerdo a lo anterior se puede decir que un ciclo simple es aquel que pasa por todos los vértices del grafo (ciudades) y que este puede ser llamado ciclo hamiltoniano o tour.
- Un subtour es también un ciclo que tiene como diferencia que este no pasa por todos los vértices (ciudades) del grafo.

Por lo tanto el problema de agente viajero, busca el tour o ciclo que implique un costo mínimo en el desplazamiento de una ciudad a otra. En base a esto existen diversas formulaciones matemáticas del problema, a continuación se muestra la formulación[6] de este problema de la siguiente manera:

Se define una variable entera X, tal que:

$$X_{i,j} = \begin{cases} 1, \text{ si se llega de la ciudad } i \text{ a la ciudad } j. \\ 0, \text{ en cualquier otro caso} \end{cases} \qquad (1)$$

$X_{i,j}$ = Variable de decisión, la cual representa la conexión o la ausencia de esta en la ruta (i,j).

Si $C_{ij,}$ es la distancia existente de ir de la ciudad *i* a la *j* y considerando que se tienen *n* cantidad de nodos, el problema tendrá como objetivo minimizar, por lo cual el modelo para el TSP es el siguiente:

---

[6] *Capítulo 1: Problema del agente viajero*. Recuperado de http://sedici.unlp.edu.ar

Minimizar

$$\sum_{i=1}^{n} \sum_{i=1}^{n} c_{ji}\, x_{ij} \quad (2)$$

donde $c_{ij} = \infty$ para $i=j$.

Sujeto a

$$\sum_{i=1}^{n} X_{ij} = 1 \quad (3)$$

$$\sum_{j=1}^{n} X_{ij} = 1 \quad (4)$$

$$X_{ij} \in \mathrm{Z}^{+} \; \forall \; \mathrm{i} \; \mathrm{y} \; \mathrm{j}$$

Las anteriores restricciones hacen referencia al número de veces que se debe llegar a una ciudad y las veces que se debe salir de ellas. La restricción (3) asegura que cada una de las ciudades 1,…. n, de salidas lleguen a una ciudad y la restricción (4), asegura que un tour cuente con una salida desde cada ciudad.

2.5 TEORÍA DE LA COMPLEJIDAD COMPUTACIONAL

La definición de la Máquina de Turing[7] en el año 1936, es una de las bases fundamentales para el desarrollo de la complejidad computacional, dado que esta entidad matemática formalizó el concepto de algoritmo y proporcionó una visión de una computadora flexible y robusta, gracias a esta máquina se demostró la existencia de problemas difíciles de resolver.

[7] Máquina de Turing, dispositivo inventado por Alan Mathison Turing (1921-1953).

Años después, en la década de los cincuenta se desarrollan los primeros lenguajes de programación y sistemas operativos, los cuales permitieron identificar que los computadores que se tenían hasta ese momento poseían recursos limitados, debido a su poca memoria para el almacenamiento de la información y la lentitud de los procesadores. Estas limitaciones eran críticas en esa época y se hicieron más evidentes en los años 60's con el origen de la *Teoría de la Complejidad Computacional* planteada por Hartmanis, P.M. Lewis y R.E Stearns[8], la cual tenía la idea o el objetivo de cuantificar el tiempo y el espacio y con base en esto se generó el análisis de los algoritmos, el cual se encarga de determinar qué cantidad de recursos son requeridos por un determinado algoritmo para resolver un problema, así mismo se planteó la clasificación de los problemas según la cantidad de recursos necesarios para encontrar su solución.

La clasificación de los problemas, ha permitido que se introduzcan nuevos modelos matemáticos con el fin de encontrar soluciones a estos de manera eficiente, cumpliendo con los recursos necesarios que requieren, como lo son el tiempo y el espacio o almacenamiento. A través de la clasificación y el análisis de los problemas surgen las *clases de complejidad*, las cuales deben poseer ciertas características necesarias para definirlas, las cuales fueron descritas por *J. Perez Jiménez y Sancho Caparrini, en su libro Máquinas moleculares basadas en ADN*, y plantean lo siguiente:

1. Un modelo de computación capaz de proporcionar dispositivos en los cuales se va a resolver los problemas.

2. Un modo de computación que se encargue de precisar el concepto de aceptación requerido para un dato de entrada.

[8] J. Perez Jiménez y Sancho Caparrini, (2003). *Máquinas moleculares basadas en ADN*, España.

3. Una medida de complejidad, la cual permita cuantificar los recursos que se emplean en los dispositivos computacionales en la solución de los problemas.

4. Una función total que delimite la cota superior de los recursos usados, la cual debe ser computable.

De acuerdo con lo anterior se han podido definir diferentes clases de problemas, según la complejidad que posee, algunas de las clases de problemas más conocidos son, clase P, NP, NP-Completos y NP-Hard.

2.5.1 CLASE P

En la teoría de la complejidad computacional la clase P es una de las clases más importantes, en esta se encuentran los problemas de decisión los cuales son tratables dado que se pueden resolver en un tiempo razonable, debido a que existe una máquina determinista secuencial que en proporción a los datos de entrada genera una solución en tiempo polinómico.

La clase P contiene todos aquellos problemas que son tratables y para los cuales existe un algoritmo de tiempo polinomial que permita solucionarlo, por lo tanto una definición de estos algoritmos es la siguiente:

Si existe un polinomio P tal que,

$$c_A(n) \leq P(n), \text{ para todo } n \qquad (5)$$

en donde,

$C_A(n)$ = Complejidad del algoritmo *A* para una entrada de tamaño *n.*

2.5.2 CLASE NP

La clase NP son aquellos problemas de decisión, en donde se encuentran los problemas relacionados con búsqueda y optimización, los cuales son resueltos en tiempo polinomial por medio de una máquina no determinista, que se encarga de determinar si existe una solución o existe una solución mejor que la conocida, un ejemplo de estos problemas es el *"problema del agente viajero".*

Los problemas que pertenecen a la clase NP tienen solución en tiempo polinomial si las máquinas en las que se son resueltos tienen la capacidad de realizar en paralelo y de manera independiente un número no acotado de computaciones.

Esto lleva a definir esta clase de la siguiente manera:

"Decimos que π pertenece a la clase NP si para cualquier instancia positiva con una entrada w del problema π existe una palabra $v(w)$ cuyo tamaño es a lo sumo $p(|\ w\ |)$ y un algoritmo que con input $(v(w),w)$ demuestra en tiempo $\leq p(|\ w\ |)$ que $w \in \pi$. Nos referimos a $v(w)$ como el certificado, es decir, se trata de la información que agregamos a la descripción del problema π que nos permite verificar que w es una instancia positiva. Más aún, decimos que el certificado es sucinto porque el tamaño de $v(w)$ es $\leq p(|\ w\ |)$"[9]. (María Lorena Stockdale, 2011, p.19).

[9] Stockdale María Lorena (2011). *El problema del viajante: un algoritmo heurístico y una aplicación*.

Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales. Departamento de matemáticas.

2.5.3 CLASE NP-COMPLETOS

La clase de los problemas de los NP-Completos son considerados los problemas más difíciles de resolver, siendo estos un subconjunto de la clase de problemas NP. La fundamentación del concepto se presentó en 1971 por Stephen Cook en el artículo "The Complexity of Theorem Proving Procedures"[10]. Esta clase de problemas ha sido objeto de estudio y ha generado debates entre investigadores y científicos debido a que se ha planteado el siguiente interrogante:

$$¿P = NP?$$

Este interrogante se resolverá en el momento en que se halle un algoritmo eficiente que encuentre una solución polinómica, dado que si se tiene una solución para esta clase de la misma manera se daría solución a la clase de problemas NP, lo cual refleja una característica de estos de equivalencia, dado que si se prueba que para los NP-Completos no existe algoritmos eficientes entonces ninguno lo tendrá.

La definición formal que plantea Luis Disset (2004) de los problemas de clase NP-Completos, los cuales, como se dijo anteriormente son problemas de decisión que toman ciertos parámetros y generan como respuesta un SI o No, es la siguiente:

"*Si $\pi = (D, L)$ es un problema de decisión, dada una instancia I de π (o sea, un elemento de D) anotaremos $I \in \pi$ para indicar $I \in L$"*

Por lo tanto un problema de decisión π tiene asociado un conjunto de instancias y subconjunto de estas en donde la respuesta es SI. Los problemas de decisión son

[10] Garey R. Michael, Johnson S. David. *Computer and Intractability. A guide to the Theory of NP-Completes*. San Francisco: W. H Freeman and Company

NP-Completos si es un NP y si todos los problemas NP se pueden reducir o demostrar que su solución se puede verificar en tiempo polinómico.

En el siguiente esquema se puede visualizar las clases NP y NP-Completo y la representación de las condiciones posibles, según los planteamientos anteriores:

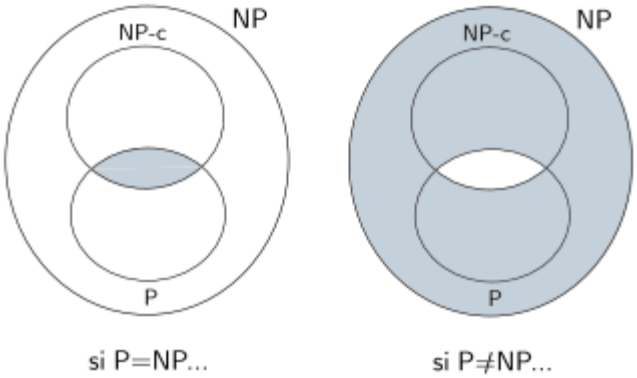

Figura 7. Clases NP Y NP-Completo

Fuente: *Complejidad - Problemas NP-Completos Algoritmos y Estructuras de Datos III*. Http://www-2.dc.uba.ar

2.5.4 Clase NP-HARD

Los problemas que pertenecen a la clase NP-Hard o NP-duros son problemas de decisión y son conocidos como los problemas más difíciles de resolver dentro de la clase NP, de los cuales aún no se tiene evidencia de la existencia de algoritmos que generen una solución en tiempo polinomial para estos.

La clase NP-Hard se define de la siguiente manera:

Se tiene un problema de decisión π, por lo tanto

$$\pi = NP - Hard \quad (6)$$

Condición 1

Si NP se puede reducir polinomialmente a π.

Por lo tanto,

Condición 2

Si π = NP-Completo y cumple con la *condición 1*

$$\pi = NP - Hard$$

3. PROBLEMA DE RUTAS DE VEHICULO (VRP)

3.1 INTRODUCCIÓN

La búsqueda de métodos y herramientas que le permitan a las empresas y organizaciones ser más competitivas y optimizar sus recursos, cumpliendo con los requerimientos de sus clientes, las cuales se encuentran enfocadas en áreas como la industria, la logística, el transporte entre otras, es uno de los temas que ha generado un alto grado de interés investigativo a lo largo de la historia y aún en el siglo XXI, dado que al lograrse establecer un modelo que permita mejorar el desempeño logístico, las empresas podrían aumentar sus ganancias y su nivel de competitividad, pues como lo expresaron Toth y Vigo (2002) *"El problema de distribuir productos desde ciertos depósitos a sus usuarios finales juega un papel central en la gestión de algunos sistemas logísticos, y su adecuada planificación puede significar considerables ahorros. Esos potenciales ahorros justifican en gran*

medida la utilización de técnicas de investigación operativa como facilitadoras de la planificación, dado que se estima que los costos del transporte representan entre el 10% y el 20% del costo final de los bienes"[11]*.* De acuerdo con lo anterior, las investigaciones y estudios llevaron al planteamiento del problema de ruteo de vehículos (VRP en ingles, Vehicle Routing Problem).

El problema de ruteo de vehículos es conocido como uno de los problemas de optimización combinatoria que pertenece a la clase NP-Hard, dado que no es posible resolverlos en un tiempo polinómico; su planteamiento se basa en encontrar un conjunto de rutas a un costo mínimo, la cual inicia y finaliza en el depósito, cumpliendo con la demanda de cada uno de los clientes a los cuales debe visitar.

Gracias a los avances de los sistemas informáticos y el desarrollo de la información geográfica, el problema de ruteo de vehículos ha podido generar soluciones adecuadas ampliando su campo de aplicación, no solo generando beneficios para las empresas, sino que además tratar problemas de transporte y de congestión vehicular que producen grandes impactos tanto en la economía de los países como en la contaminación ambiental. En esta sección se abordarán los diferentes tipos de problema de ruteo de vehículos (VRP) existentes, el origen y su planteamiento.

[11] Toth Paolo, Vigo Daniele (2002). *The vehicle routing problem.*

3.2 HISTORIA

Los inicios del problema de ruteo de vehículos (VRP) se pueden establecer con el surgimiento del problema del agente viajero en el año 1956, dado que la formulación planteada por Flood, permitió que surgieran variaciones de este, una de ellas fue establecida por Dantzing y Ramser en el año 1959, en su libro *"The truck dispatching problem"*[12], en el que se planteó un modelo de despacho de combustible el cual se realizaba a través de una flota de camiones a diferentes estaciones de servicio, dando surgimiento al concepto del problema de ruteo de vehículos.

El planteamiento del agente viajero no solo permitió que surgiera el problema de ruteo de vehículos sino que además fundamentó las bases de nuevos problemas que han sido objeto de estudio a lo largo de la historia, tales como:

- TSP múltiple o m-TSP planteado en 1960 por Miller, Tucker y Zemlin.

- TSP probabilístico o PTSP planteado por Tillman en 1969.

En el año de 1981 Lenstra y Rinnooy Kan[13], analizaron la complejidad del problema de ruteo de vehículos, en donde concluyeron que estos problemas pertenecen a la clase NP-Hard, dado que no se pueden resolver en tiempo polinomial.

---

[12] Dantzig G. B. y Ramser J. H. (Oct. 1959). *The truck Dispatching Problem.* (Ed. 6)

[13]Caric Tonci y Gold Hrvoje (2008). *Vehicle Ruting Problem*. Croacia: In-The.

3.3 DEFINICIÓN

El problema de ruteo de vehículos (VRP) es un problema de optimización combinatoria perteneciente a la clase NP-Hard con aplicaciones en la práctica logística, el cual tiene como objetivo reducir el coste total de una ruta. El planteamiento del problema consiste en la existencia de un depósito central que cuenta con una flota de vehículos y un conjunto de ciudades o clientes que se encuentran dispersos geográficamente, se debe encontrar un conjunto de rutas a un costo mínimo, visitando cada cliente una sola vez y con un solo vehículo, el cual tiene una capacidad limitada.

El problema de ruteo de vehículos está compuesto por clientes, depósitos y vehículos, los cuales según sus características puede generar variantes de este problema.

- Los clientes poseen una demanda que deberá ser satisfecha, la cual consiste en que debe ser visitada al menos una vez por los vehículos; además posee restricciones relativas a los horarios de servicio o a la compatibilidad con los vehículos (es decir, que sólo podrá ser visitado por ciertos vehículos, en aquellos casos en donde se empleen más de un vehículo para hacer los recorridos).

- Los depósitos poseen diferentes características como por ejemplo, la ubicación y la capacidad de producción, estos al igual que los clientes poseen restricciones, de las cuales, las más usuales son las relacionadas con los tiempos empleados en preparar a los vehículos para que inicien su recorrido o el tiempo que deben invertir en regresar después de cumplir con la ruta establecida. Generalmente se establece que cada ruta inicie y finalice en un mismo depósito.

- Los vehículos tienen atributos como capacidad y costo, cuando se habla de capacidad, estos pueden tener varias dimensiones, como por ejemplo peso y volumen, por lo tanto, cuando se hace referencia al costo, los vehículos pueden

tener asociados costos fijos y costos variables, siendo estos últimos proporcionales a la distancia que recorran. Cuando estos atributos son iguales para todos los vehículos se tiene una *flota homogénea* y cuando hay diferencias se denomina *flota heterogénea*. Dentro de las restricciones que tienen los vehículos se encuentran las relacionadas con el tiempo, las cuales estipulan un tiempo máximo de circulación del vehículo, y las restricciones relacionadas con los clientes, en las que se establecen las zonas a las cuales no podrá visitar.

En la figura 8 se puede observar el funcionamiento del problema de ruteo de vehículos y cada uno de los componentes anteriormente descritos.

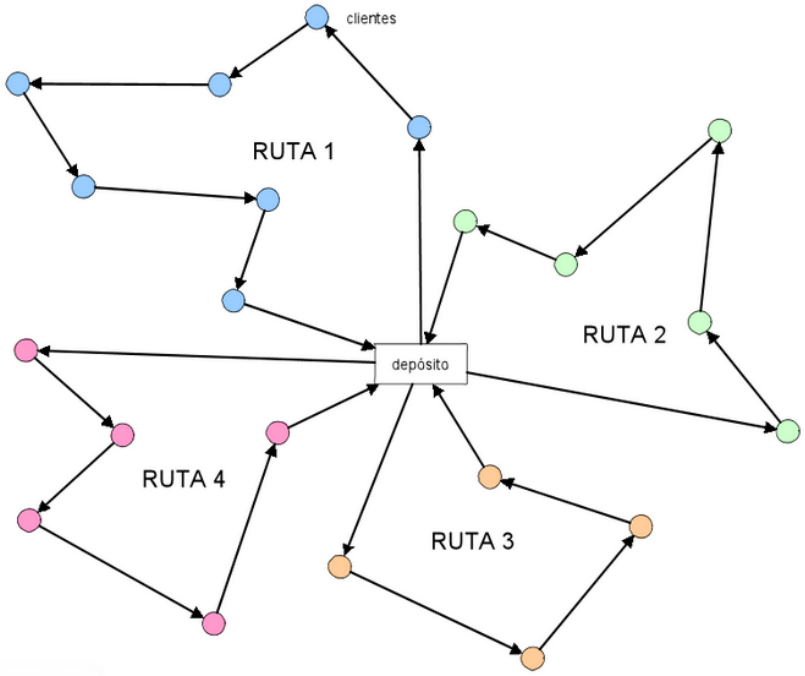

Figura 8. Representación problema de ruteo de vehículos

Fuente: Caric Tonci y Gold Hrvoje (2008). *Vehicle Ruting Problem*. Croacia: In-The.

3.4 TIPOS DE VRP

El problema de ruteo de vehículos VRP tiene un campo de aplicación muy amplio, lo que permite que este pueda variar de acuerdo a las necesidades que se presenten, es por esta razón que los atributos y restricciones que se emplean al momento de plantear diferentes problemas, generan que el VRP tenga nuevos enfoques y que de acuerdo al empleo de distintos atributos se creen nuevos tipos o clases de este problema, permitiendo que se dé una mejor búsqueda de las soluciones, en donde los detalles o tomas de decisiones sean más precisas.

Existen diferentes tipos de problemas de ruteo de vehículo, dentro de los cuales se destacan los siguientes, los cuales se visualizan en la figura 9.

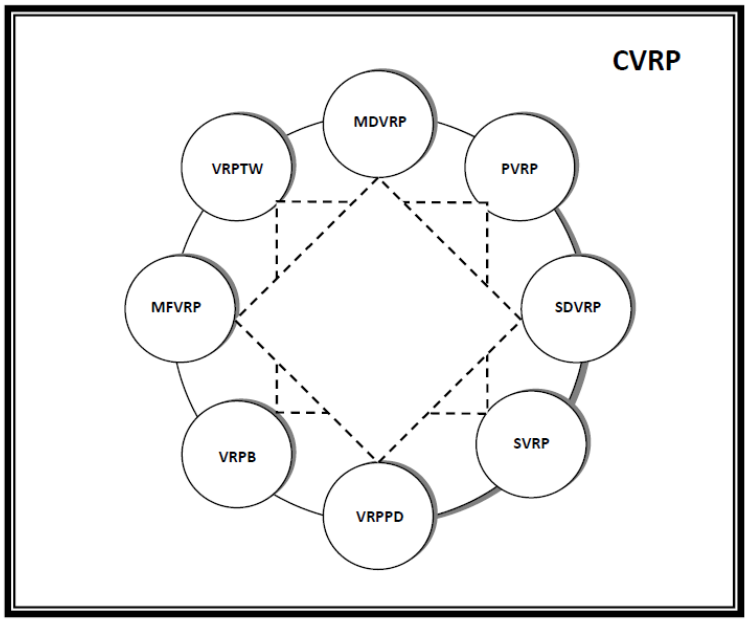

Figura 9. Variantes del VRP

Fuente: Gonzales et al. (2006).

Problema con restricciones de Capacidad (CVRP)

El problema CVRP (*Capacited VRP*) es una variante del VRP que consiste en que dadas *n* ciudades, en donde se cuenta con un depósito único y una flota de vehículos que emplean habitualmente capacidad fija y que deben prestar un servicio al número de clientes desde el depósito central a un costo mínimo, se debe encontrar los recorridos de los vehículos de tal manera que se minimice la distancia y se satisfaga la demanda de los cliente cumpliendo con la restricción de no exceder la capacidad del vehículo.

Problema con múltiples depósitos (MDVRP)

El problema con múltiples depósitos conocido como MDVRP (por sus siglas en ingles, Multiple Depot VRP), tiene como objetivo brindar un servicio a todos los clientes minimizando el número de vehículos y la distancia recorrida por ellos. En este problema se tiene la variante de que existen varios depósitos desde donde se puede ofrecer el servicio a los clientes, cada uno de estos dispone de su propia flota de vehículos, los cuales deben salir del depósito realizar la ruta correspondiente y regresar a su depósito de origen.

Problema periódico (PVRP)

El PVRP (Periodic VRP) tiene como objetivo minimizar la flota de vehículos y los tiempos de recorrido empleados para brindarles el servicio a los clientes. Su planteamiento consiste en establecer un horizonte de operación M días, en este periodo se debe visitar al menos una vez a cada cliente.

Problema de entregas parciales (SDVRP)

El problema SDVRP (Split Delivery VRP) o también conocido como VRP de entrega divida, consiste en que un cliente podrá ser visitado por diferentes vehículos reduciendo los costos, siempre que la demanda del cliente sea mayor a la capacidad de los vehículos.

Problema con valores al azar (SVRP)

El problema SVRP (Stochastic SVRP) tiene como objetivo minimizar la distancia recorrida por el vehículo de manera que considerando la aleatoriedad de las variables se pueda cumplir al cliente con sus demandas, servicios y tiempos de viaje. Este problema considera uno o varios componentes aleatorios, como por ejemplo el número de clientes, tiempo de recorrido, etc. Existen tres casos para este tipo de problema los cuales son:

- Clientes estocásticos

-Demandas estocásticas

-Tiempos estocásticos

Problema con entregas y devoluciones (VRPPD)

El problema VRPPD (VRP with Pickup and delivery) plantea de que existe la posibilidad que los clientes puedan devolver algún tipo de mercancía, por lo que se debe tener en cuenta que estos artículos si quepan en el vehículo y no excedan su capacidad. El objetivo de este problema consiste en minimizar la flota de vehículos y la suma de los tiempos de transporte bajo la restricción de que cada uno de los vehículos debe contar con la capacidad suficiente para transportar los productos que deberán recogerle a los clientes con el fin de regresarlos al depósito.

Problemas con viajes de regreso (VRPB)

El problema VRPB (VRP with Bachauls) tiene un planteamiento similar al VRPPD que consiste en que los clientes pueden tanto recibir mercancías o productos como entregarlos, pero este tiene la variante de que existe una restricción en la que las entregas deben ser realizadas y completadas antes de que se realicen las devoluciones.

Problema con múltiples depósitos (MFVRP)

El problema MFVRP (Mixed Fleet VRP) consiste en que los vehículos pueden tener distintas capacidades, por lo cual se deberá determinar que vehículo será el adecuado para realizar el recorrido de una ruta según la distancia y la demanda.

Problema con ventanas de tiempo (VRPTW)

El problema VRPTW (VRP with time windows) tiene el mismo planteamiento del VRP pero con la restricción de que existe un periodo de tiempo determinado para cumplir con el abastecimiento de los clientes.

En la tabla 2 se muestran otras de las variantes del problema de ruteo de vehículos VRP.

Tabla 2. Tipos de VRP

Siglas	Nombre	Particularidad
AVRP	Asymmetric Vehicle Routing Problem	La distancia recorrida en el viaje o el tiempo necesario entre un par de puntos depende del sentido del trayecto
CVRP	Capacitated Vehicle Routing Problem	El vehículo tiene una capacidad de carga o transporte limitada, que supone una restricción que no debe ser superada.
DCVRP	Distance Constrained Vehicle Routing Problem	VRP con limitación en la distancia total recorrida o en el número de clientes visitados.
DVRP	Dynamic Vehicle Routing Problem	Problemas en los cuales algunos parámetros dependen del tiempo, o cambian dinámicamente en el tiempo
FRP	Fixed Routes Problem	Una vez determinadas las rutas, éstas no deben variar durante un periodo de tiempo, aunque así lo haga la demanda.
FSMVRP	Fleet Size and Mix Vehicle Routing Problem	Flota de vehículos ilimitados, con costes fijos según el tipo de vehículo y costes variables homogéneos.
LVR	Location Routing Problem	En este problema se intenta determinar la localización del depósito que no es conocida a priori.
MCVRP	Multi Compartment Vehicle Routing Problem	Los vehículos están configurados mediante compartimentos. Su carga es mixta y las mercancías deben permanecer separadas durante el viaje.
MDVRP	Multiple Depot Vehicle Routing Problem	No sólo hay un único depósito, sino que existen varios (origen y destino final de sus vehículos asignados).
min-max VRP	Min-max Vehicle Routing Problem	La función objetivo es minimizar la distancia del trayecto más largo
OVRP	Open Vehicle Routing Problem	Algunos vehículos no tienen porqué volver al depósito (transporte subcontratado).
PVRP	Period Vehicle Routing Problem	Cada cliente requiere un número determinado de servicios en un periodo de tiempo.
RTDVRP	Real Time Dynamic Vehicle Routing Problem	Conjunto de problemas donde las condiciones dinámicas obligan a la reprogramación de servicios en tiempo de ejecución de la ruta.

Siglas	Nombre	Particularidad
SVRP	Stochastic Vehicle Routing Problem	Conjunto de problemas donde algunos parámetros tienen cierto grado de incertidumbre. Dentro de este conjunto destacan los dos siguientes:
VRPSD	Vehicle Routing Problem with Stochastic Demands	Los clientes pueden presentar demandas aleatorias.
VRPSDC	Vehicle Routing Problem with Stochastic Demands and Customers	Tanto la presencia de clientes como su demanda son aleatorias.
VFMVRC	Vehicle Fleet Mix with Variable Unit Running Costs	Costes fijos y variables dependientes del tipo de vehículo. Número de vehículos ilimitados.
VRPB	Vehicle Routing Problem with Backhauls	Existen puntos de entrega desde el almacén y otros de recogida hacia el almacén. No se recoge hasta que no finalicen las entregas.
VRPDB	Vehicle Routing Problem with Deliveries and Backhauls	Existen puntos de entrega y otros de recogida hacia el almacén, pudiendo coincidir en ambos. Se permite la entrega y recogida mientras no se viole la capacidad del vehículo.
VRPHE	Vehicle Routing Problem with Heterogeneous Fleet	Costes fijos y variables que dependen del tipo de vehículo. Número limitado de vehículos de cada tipo.
VRPLC	Vehicle Routing Problem with Length Constraint	La longitud de cada ruta no debe superar una magnitud determinada. Alternativamente se puede limitar el tiempo o el número de clientes visitados. Mismo significado que DCVRP
VRPM	Vehicle Routing Problem with Multiple Use of Vehicles	Cada vehículo puede emprender más de una ruta en un periodo de tiempo
VRPPC	Vehicle Routing Problem with Precedent Constraints	Existen relaciones de precedencia entre clientes, por lo que antes de visitar a un cliente, el vehículo debe visitar previamente a un conjunto de ellos.
VRPPD	Pickup and Delivery Problem	Un vehículo debe recoger la mercancía en un sitio y llevarla a otro de la red.

Siglas	Nombre	Particularidad
VRPSD	Vehicle Routing Problem with Split Delivery	La demanda de un cliente puede ser cubierta por varios vehículos. El servicio se fragmenta entre varios vehículos o rutas. También se conoce como SDVRP.
VRPSF	Vehicle Routing Problem with Satellite Facilities	Existen depósitos intermedios donde pueden reabastecerse los vehículos sin necesidad de volver al depósito central.
VRPST	Vehicle Routing Problem with Stochastic Travel Times	La duración de los viajes tiene un carácter aleatorio.
VRPSTW	Vehicle Routing Problem with Soft Time Windows	VRP con ventanas horarias donde se permite cierta trasgresión del horario de entrega de cada cliente e incluso del depósito mediante penalizaciones.
VRPTD	Vehicle Routing Problem with Time Deadlines	Las ventanas de horario de entrega sólo tienen su limitación final.
VRPTW	Vehicle Routing Problem with Time Windows	Cada cliente presenta una (o varias VRPMTW) ventana horaria de reparto o entrega. El depósito también tiene un horario de disponibilidad.
VRPVADT	Vehicle Routing Problem with Variable Access Time	La duración del acceso al cliente y la salida dependen del cliente.
VRPVRT	Vehicle Routing Problem with Variable Travel Times	La duración de los viajes es variable, y depende del horario en el que se realicen.

Fuente: Pérez Rodríguez Jorge. *Caracterización, modelado y determinación de las rutas de flota en una empresa de Rendering (p.16).*

3.5 FORMULACIÓN DEL PROBLEMA

El problema de ruteo de vehículos se puede plantear de la siguiente manera:

Sea

C, es la capacidad de los vehículos,

m, el número de vehículos

n, el número de clientes

d, demanda de la mercancía

Minimizar

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} C_{ij} \left(\sum_{k=1}^{m} X_{ij} \right) \tag{6}$$

Sujeto a las siguientes restricciones;

$$\sum_{j=2}^{n} X_{1j} \leq m \tag{7}$$

No se permiten la salida de más vehículos de los que se tienen.

$$\sum_{i=2}^{n} X_{i1} = \sum_{j=2}^{n} X_{1j} \tag{8}$$

El número *m* de vehículos que salen del punto 1, son los mismos que regresan.

Sea

u_i y u_j variables externas auxiliares

Q, la demanda total

Se deberá cumplir con la capacidad máxima y evitar que se generen subciclos

$$u_i - u_j + QX_{ij} \leq Q - d_i, \forall i \neq j, i, j \in \{2, \dots, n\}, X_{ij} \in \{0,1\}, u \in R^+ \tag{9}$$

El problema de ruteo de vehículos es modelado mediante un grafo, esta representación matemática planteada por Toth y Vigo[14] (2002) es la siguiente:

"La red de transporte por la que circulan los vehículos se modela mediante un grafo ponderado *G = (V, E,* C). Los nodos del grafo representan a los clientes y depósitos. En problemas con un depósito y *n* clientes, el nodo {0} representa al depósito y los nodos {1,….,n} a los clientes. En algunos casos se agrega una copia del depósito etiquetada con n+1 para simplificar la formulación.

Cada arco *(i, j)* $\in E$ representa el mejor camino para ir desde el nodo i hacia el nodo j en la red de transporte y tiene asociado un costo c_{ij} y un tiempo de viaje t_{ij}. Según la estructura de los costos y los tiempos y las características de la red, el grafo puede ser simétrico o asimétrico. Puede suponerse que G es completo, pues entre todo par de lugares de una red de transporte razonable, debería existir algún camino. Sin embargo, por una cuestión de flexibilidad, los modelos serán planteados sin realizar dicha hipótesis.

Denotaremos por $\Delta^+(i)$ y $\Delta^-(i)$ al conjunto de nodos adyacentes e incidentes al nodo *i*, es decir, $\Delta^+(i) = \{j \in V \mid (i,j) \in E\}$ y $\Delta^-(i) = \{ j \in V \mid (j, i) \in E\}$.

De manera similar, el conjunto de arcos incidentes hacia el exterior e interior del nodo *i* se definen como

$$\delta^+(i) = \{(i.j) \in E\}\, y\, \delta^-(i) = \{(j,i) \in E\}\text{".} \qquad (10)$$

[14] P. Toth, D. Vigo, *The Vehicle Routing Problem. Monographson Discrete Mathematics and Applications*. Society of Industrialand Applied Mathematics. Philadelphia. USA, 2002

4. ALGORITMO GÉNETICO DE CHU-BEASLEY (AGCB)

4.1 INTRODUCCIÓN

En la búsqueda de dar solución a aquellos problemas que son clasificados como difíciles de resolver, se han empleado métodos metaheurísticos para este tipo de problemas, tales como el algoritmo genético de Chu-Beasley (AGCB), el cual es una versión transformada de un algoritmo genético simple, con algunas diferencias que hacen de este un método eficiente en el momento de evaluar sistemas y resolver problemas de gran tamaño y complejidad.

El algoritmo genético de Chu-Beasley (AGCB) se desarrolla mediante el proceso evolutivo clásico, en el cual se realiza la selección, la recombinación y la mutación, buscando mantener la diversidad de los individuos de una población, esta diversidad se logra gracias a que el AGCB en cada generación reemplaza un solo individuo de la población, verificando que se cumplan las condiciones de diversidad, factibilidad y optimalidad establecidas; a diferencia de los algoritmos genéticos tradicionales, que emplean su proceso de sustitución a todos o a casi todos los individuos de una población en donde generalmente no se logra tener mayor diversidad genética. De acuerdo con lo anterior, el algoritmo de Chu-Beasley, mediante la sustitución de uno de los individuos de la población, genera una mejor descendencia mediante una optimización local, permitiendo un control mayor sobre la diversidad de la población.

Dado que el algoritmo genético de Chu-Beasley (AGCB) pertenece al grupo de las metaheurísticas, genera soluciones a problemas de optimización para los cuales las técnicas exactas no son eficientes o no es posible que se puedan aplicar, por tal razón las técnicas metaheurísticas comparadas con las heurísticas permiten encontrar la solución óptima global, aunque ninguno de los algoritmos

pertenecientes a esta técnica garantiza que se dé a los problema el óptimo global, pero sus soluciones son superiores a las generadas por las heurísticas.

El algoritmo genético de Chu-Beasley presenta un mejor desempeño en el momento de encontrar las soluciones y como se ha expresado anteriormente, esto se debe a su enfoque en la fase de mejora local de la descendencia, la cual es una sencilla búsqueda local o una estrategia sofisticada que tiene presente las características especificas del problema que se está considerando. Esta fase se divide en dos sub-fases, en donde la primera tiene un enfoque de mejora de inviabilidad y la segunda de mejora de calidad. Por esta razón como la fase de mejora local trae consigo la sustitución de un individuo de la población, busca preservar la diversidad, por lo tanto si la descendencia que se ha generado de la población inicial es igual se descarta, en caso contrario se establecen dos procesos basados en las sub-fases mencionadas, en donde si la descendencia es considera inviable se realiza la verificación de sí la inviabilidad es más pequeña que la inviabilidad que posee un elemento de la población tenía la mayor inviabilidad, se realiza la sustitución de los individuos, en caso contrario se descarta la descendencia generada, cumpliendo de esta manera con la primer fase. Si se tiene el caso de que la descendencia que se ha generado es factible se realiza la sustitución del individuo que posea la mayor inviabilidad, si se da que todos los elementos de la población son factibles, la sustitución solo es posible si al verificar que la descendencia generada presenta mejor calidad que el elemento que tenga la calidad más baja de la población inicial.

Por tal motivo la propuesta de sustitución que presenta el algoritmo genético de Chu-Beasley permite que un control absoluto de la diversidad de la población lo que genera una gran ventaja para aquellos problemas que presentan grandes dificultades a la hora de encontrar soluciones viables, debido a que este algoritmo garantiza que se verifica tanto la inviabilidad de la descendencia y de los

individuos como de la calidad que se encarga de determinar la factibilidad de las propuestas de solución. En esta sección se abordará los aspectos más importantes del AGCB y su funcionamiento.

4.2 DEFINICIÓN

El algoritmo genético de Chu-Beasley (AGCB) se basa en la teoría fundamental de los algoritmos genéticos, diseñado inicialmente para la solución de problemas de designación generalizada, pero con base a las diferencias que presenta, en comparación con los algoritmos genéticos simples, se empleó en la solución de problemas de gran tamaño y complejidad tales como los que pertenecen a la clase NP-Hard.

El AGCB presenta una característica particular que lo hace más eficiente y permite que las soluciones no caigan en óptimos locales, la cual consiste en mantener durante todo el proceso evolutivo (que comprende la selección, la recombinación y la mutación) la diversidad de los individuos de la población; esta diversidad se logra sustituyendo o reemplazando solo un individuo de la población, siempre y cuando cumpla con las condiciones establecidas de optimalidad, diversidad y/o factibilidad.

El algoritmo genético de Chu-Beasley al ser una "*versión modificada*" de los algoritmos genéticos tradicionales presenta modificaciones y características particulares como:

1. La función objetivo es empleada para realizar la identificación de los individuos de mejor calidad manejando la infactibilidad.

2. Reemplazo de un solo individuo de la población en cada ciclo generacional, diferenciándose del planteamiento propuesto por Holland sobre el algoritmo genético simple.

3. Es un algoritmo elitista, dado que solo será reemplazado a la siguiente generación un individuo padre, por un descendiente hijo si este último tiene una función de ajuste de mejor calidad que la del padre. Es decir que contiene un criterio de aspiración o aceptación, en donde el individuo nuevo puede ingresar a la población aunque no cumpla con los criterios de diversidad establecidos, siempre y cuando este sea la mejor solución encontrada hasta el momento.

4. Posee el criterio de diversidad de individuos que evitan la convergencia prematura de las soluciones a óptimos locales.

5. Tiene una fase mejora de la descendencia, después de realizar los procesos de selección, recombinación y mutación, permitiendo realizar un análisis y explotar la solución descendiente antes de determinar si se realiza el reemplazo del individuo de la actual población.

El algoritmo genético de Chu-Beasley realiza los procesos evolutivos llevados a cabo por los algoritmos genéticos tradicionales, los cuales se describen a continuación:

- El proceso de *selección* en los AGCB, se realiza mediante la selección de dos cromosomas padres encargados de realizar la transferencia de genes a la siguiente generación.

- La *recombinación* se realiza en un punto, en la cual los cromosomas de los padres son compartidos para generar nuevos individuos que serán candidatos para ser elegidos. Este proceso controlará la factibilidad de los individuos y al final del proceso sólo el nuevo individuo con mejor función objetivo seguirá el proceso.

- La *mutación* se lleva a cabo mediante la alteración de alguno de los alelos de forma aleatoria en donde se controla nuevamente la factibilidad.

Una vez finalizados estos procesos genéticos se realiza el proceso de aceptación en donde se evalúa cual individuo es mejor que el individuo de peor calidad, y se reemplaza, además de realizarse la evaluación de la diversidad de los individuos en la población.

4.3 COMPONENTES

El algoritmo genético de Chu-Beasley al igual que los algoritmos genéticos simples poseen los mismos componentes pero con algunas variaciones que permite que este algoritmo sea más eficiente en el momento de tratar problemas de alta complejidad, estos componentes se describen a continuación.

Codificación

El AGCB permite que se realice una codificación con variables binarias, enteras o reales, dependiendo de la naturaleza de las variables y del problema que se esté aplicando. Un ejemplo de codificación de un problema se muestra en la figura 10.

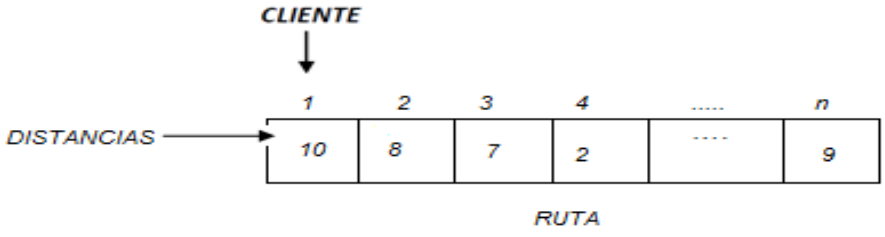

Figura 10. Ejemplo de codificación

Población Inicial

La generación de la población inicial se puede realizar de manera aleatoria o empleando algoritmos inicializadores, lo cual de acuerdo al problema que se esté tratando puede influir en la obtención de soluciones de alta calidad. En aquellos problemas en donde la complejidad es mayor el uso de los algoritmos inicializadores resulta ser más efectivo y para aquellos de complejidad baja o mediana la generación inicial se podrá efectuar de manera aleatoria permitiendo que se alcance una solución óptima.

Selección

En el proceso de selección para los algoritmos genéticos, existen diferentes métodos como por ejemplo la ruleta y torneo; en muchas de las implementaciones conocidas del algoritmo genético de Chu-Beasley emplean el método de selección por torneo. Un ejemplo del planteamiento de este método realizado por Ruiz Andrés, Toro Mirledy y Salazar Harold[15] es el siguiente:

"*La selección por torneo consiste en escoger la alternativa que tenga mejor función objetivo de los k cromosomas seleccionados aleatoriamente. El éxito de aplicar esta metodología radica en escoger adecuadamente un valor de k que se ajuste a cada problema en particular teniendo en cuenta tamaño y complejidad de la aplicación. Un valor de k muy alto puede eventualmente hacer caer el proceso en óptimos locales de muy baja calidad. El valor de k puede ser variable dependiendo el tamaño de la población; un valor recomendado de k es 2. En problemas con población grande el valor de k podrá ser mayor o igual a 2*".

[15] Ruiz Andrés, Toro Mirledy y Salazar Harold (2007a*). ALGORITMO GENÉTICO MODIFICADO CHU-BEASLEY APLICADO A LA IDENTIFICACIÓN DE ERRORES EN LA ESTIMACIÓN DE ESTADO EN SISTEMAS.* (p.28)

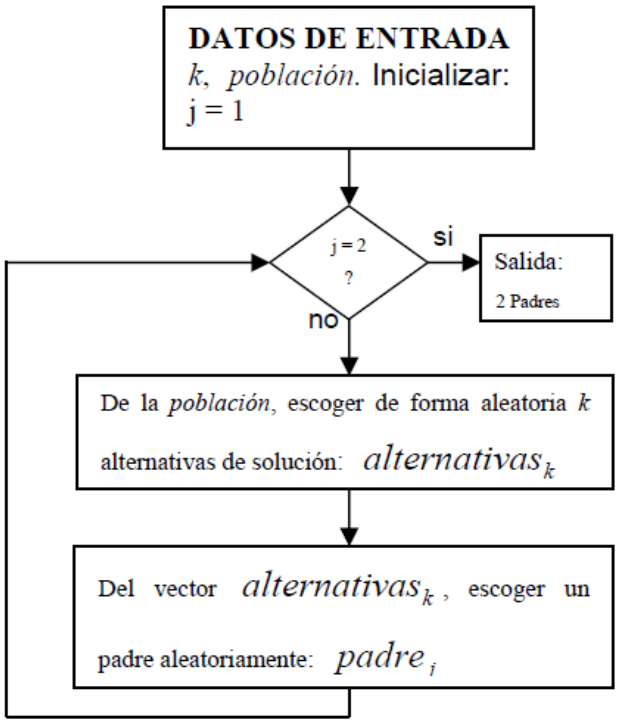

Figura 11. Proceso de selección

Fuente: Ruiz Andrés, Toro Mirledy y Salazar Harold (2007*). ALGORITMO GENÉTICO MODIFICADO CHU-BEASLEY APLICADO A LA IDENTIFICACIÓN DE ERRORES EN LA ESTIMACIÓN DE ESTADO EN SISTEMAS.* (p.28)

Recombinación

El proceso de recombinación se lleva a cabo una vez se han seleccionado los dos padres, con el fin de combinarlos y tener una generación a partir de ellos. Los puntos de recombinación se establecen de manera aleatoria para realizar la combinación de las características de los padres realizando el cruce entre las porciones de los cromosomas que se encuentran dentro de los puntos. Este proceso permite que los individuos que son descendientes cumplan con los

criterios de optimalidad y/o factibilidad establecidos además de ser una estrategia eficiente de búsqueda local.

Mutación

El proceso de mutación llevado a cabo en el algoritmo genético de Chu-Beasley realiza un reemplazo o sustitución de uno de los individuos, buscando que la diversidad de la población se mantenga. Una mutación drástica podría generar que el proceso se aleje de las zonas de búsqueda actual y se dirijan a regiones distantes determinando una especie de diversidad, pero si ocurre que la mutación que se realiza es débil puede ocasionar que la convergencia de las regiones sea local y no se dé una solución eficiente.

Proceso de aspiración

Este proceso inicia validando la diversidad de la población, ya que se emplea para aceptar o rechazar a un descendiente para que forme parte de la población. El proceso de aspiración realiza una verificación para que no haya individuos idénticos dentro de la población, al individuo descendiente, los criterios que se establecen para realizar este proceso dependen del problema y de la codificación que se plantee.

Los criterios de diversidad que se plantean en este proceso son los siguientes: la verificación de que el nuevo individuo posee la mejor función objetivo que el individuo de peor calidad, por lo tanto se reemplaza el individuo de peor calidad de la población. La comparación de un valor aleatorio con la tasa de aceptación, en donde, si es menor se le permite pasar a la siguiente generación, en caso de que no se cumpla esto, no se le permitirá ingresar a la población.

4.4 FUNCIONAMIENTO DEL ALGORITMO

El algoritmo genético de Chu-Beasley (AGCB) tiene un funcionamiento similar a los algoritmos genéticos simples, llevando a cabo los mismos procesos (selección, recombinación y mutación), pero con algunas variantes. Este algoritmo funciona de la siguiente manera[16]:

1 Se especifican los parámetros de control (tamaño de la población, la tasa de recombinación, las tasas de mutación, etc.).

2. Se especifica las características genéticas del algoritmo, tales como tipo de codificación, montaje inicial de la población, forma de selección, etc.

3. Se generar una población inicial

4. Se obtienen dos individuos de la población "*padre"* empleando selección por torneo de la población actual.

3. Se aplica recombinación a los padres seleccionados anteriormente, en donde se obtiene una alternativa *"hijo.*

7. Se obtiene una alternativa modificada aplicando *Mutación* siempre y cuando el parámetro obtenido lo permita.

8. Para modificar la población se propone la siguiente estrategia:

a. Si la alternativa actual es infactible y a su vez es menos infactible que la peor infactible de la población, entonces reemplazar la peor infactible por la alternativa actual.

---

[16] Ruiz Andrés, Toro Eliana, Salazar Harol(2007b). *Algoritmo genético modificado de Chu-Beasley aplicado a la identificación de errores en la estimación de estado en sistemas eléctricos*.(p.29)

b. Si la configuración es factible y existe por lo menos una infactible en la población actual, entonces reemplazar la peor infactible por la alternativa actual.

c. Si la configuración es factible y todas las alternativas de la población actual son factibles, entonces reemplazar la alternativa con peor función objetivo por la alternativa actual. Lo anterior se realiza sólo si la alternativa actual es de mejor calidad que l peor de la población.

7. En caso contrario desechar la alternativa resultante y volver al paso 2.

8. El proceso termina hasta cumplir el criterio de parada.

.En la figura 12, se muestra el esquema del funcionamiento del algoritmo genético de Chu-Beasley.

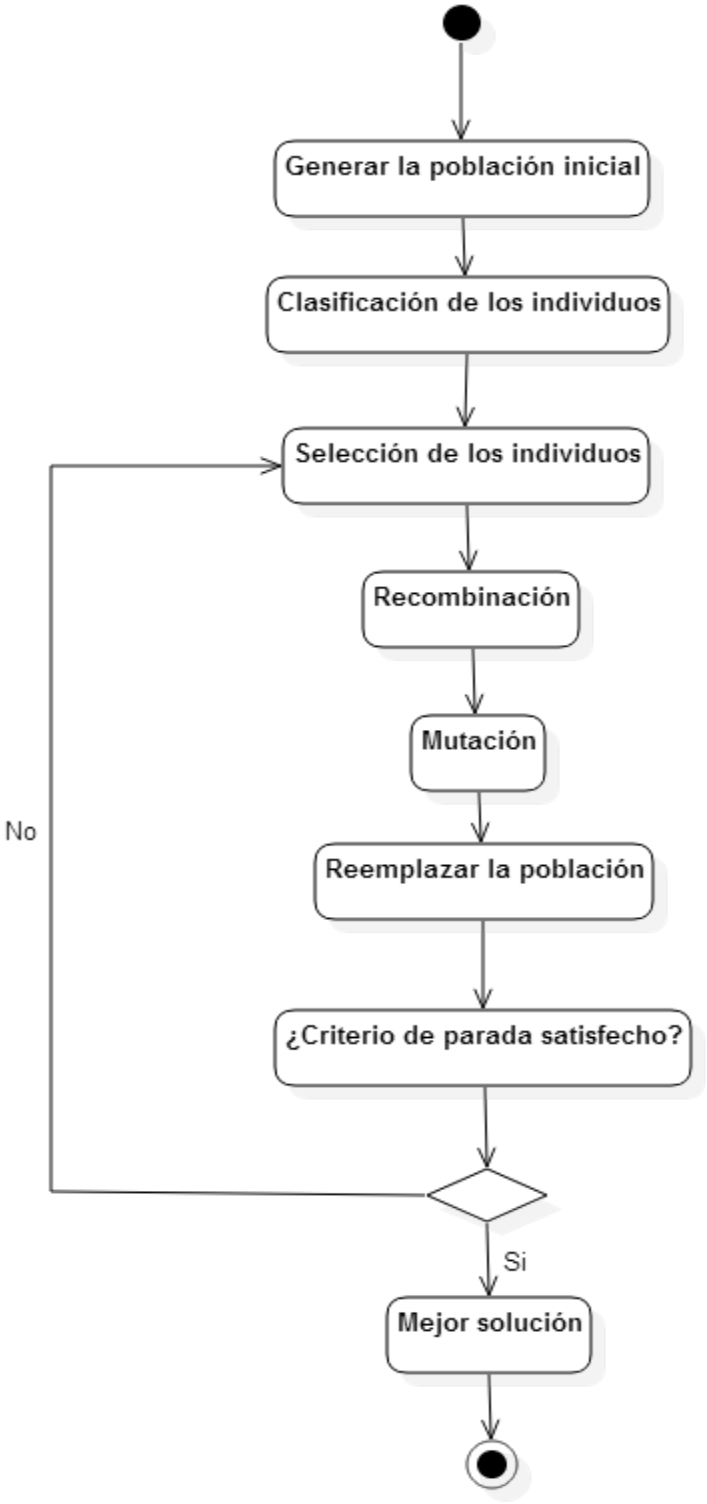

Figura 12. Funcionamiento del Algoritmo de Chu-Beasley

5. PROBLEMA DE LA MOCHILA

5.1 INTRODUCCIÓN

El problema de la mochila (en inglés, *Knapsack Problem*) es un problema de optimización combinatoria, cuya formulación es muy sencilla, aunque su resolución es compleja. Este problema ha sido objeto de estudio durante más de un siglo, ya que se hizo referencia a él por primera vez en el año 1897 en el artículo de George Mathew Ballard[17] y en las obras del matemático Tobías Dantzing. El problema de la mochila es considerado uno de los *21 problemas NP-Completos* de Richard Karp[18] que se establecieron en un artículo en el año 1972.

El problema de la mochila es un problema de tipo NP-Completo, en el cual, el tamaño y la complejidad aumentan a medida que aumenta el número de variables, por lo tanto al pertenecer a esta clase es improbable que se encuentre un algoritmo que pueda darle solución en tiempo polinomial, además este problema al no ser de tipo NP-Hard podría ser resuelto en tiempo seudo-polinomial a través de programación dinámica, lo cual indica que la codificación uniaria del problema puede ser resoluble polinomialmente.

El problema de la mochila tiene una gran variedad de aplicaciones, dentro de las que se incluyen la planificación de la producción, modelización financiera, muestreo estratificado, entre otras; abarcando áreas como la informática, la teoría de la complejidad, la criptografía y las matemáticas aplicadas. Este problema también ha sido de gran importancia, pues se ha empleado como un campo de

---

[17] G.B. Mathews. *On the partition of numbers, Proceedings of the London Mathematical Society*,

[18] Karp Richard M. (1972). Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*.

Nueva York: Plenum. (pp. 85–103).

pruebas para realizar ensayos sobre la eficiencia de métodos de solución y de búsqueda inteligente en problemas de optimización combinatoria.

Este problema tiene un interés particular debido a sus características combinatorias y a la sencillez de su estructura, que lo hacen un problema ideal para el diseño de métodos de búsqueda inteligente permitiendo evaluar las ventas y desventajas de estos algoritmos en cuanto a su robustez, precisión y rapidez. En esta sección se tratará de manera detallada la formulación y el planteamiento que tiene este problema, así como su definición y la complejidad que posee.

5.2 DEFINICIÓN

El problema de la mochila es un problema de optimización combinatoria que bajo un esquema mono-objetivo tiene como finalidad determinar cuáles elementos se deben adicionar en una mochila con el propósito de obtener el máximo beneficiando garantizando que no se exceda el volumen o capacidad de la mochila. De acuerdo con este planteamiento, se puede considerar que dado un conjunto de elementos, en el que cada uno de ellos tiene asociado una masa y un valor, se deberá determinar el número de cada elemento a incluir en una mochila o recipiente de tal forma, que el peso total sea menor que o igual a un límite dado y el valor de este sea tan grande como sea posible.

5.3 FORMULACIÓN MATEMÁTICA

Los datos del problema de la mochila se pueden expresar matemáticamente de la siguiente manera:

- La variable *p* representa el peso
- La variable *v* representa el valor
- La variable *i* son los objetos los cuales van variando de *1 a n*
- La capacidad de la mochila está dada por la variable *c*
- La función objetivo está dada por z(x)

Con el fin de decidir qué elementos se deben meter en la mochila o no, se emplea el código binario en donde x_i=1, si el objeto se mete dentro de la mochila o x_i=0 si el objeto se deja por fuera de la mochila.

Para llenar la mochila se empleará un vector de contenido *x = (x1, x2, ….xn)*

Entonces:

Para un contenido x, el valor total de la bolsa estará dado por:

$$z(x) = \sum_{\{i,x_i=1]} v_i = \sum_{i=1}^{n} x_i v_i \qquad (11)$$

La suma de los pesos estará dada por:

$$p(x) = \sum_{\{i,x_i=1]} p_i = \sum_{i=1}^{n} x_i p_i \qquad (12)$$

El problema por lo tanto se planteará de la siguiente manera:

Se tiene un vector $x = (x1, x2, \ldots . xn)$ que tiene componentes de unos y ceros, en donde se debe tener en cuenta la restricción de la función z(x), lo cual quiere decir

que la suma de los pesos p(x), de los objetos que se metieron en la mochila no pueden superar la capacidad de esta, la cual está dada por *c*.

$$p(x) = \sum_{i=1}^{n} x_i \, p_i \leq c \qquad (13)$$

Restricciones:

1. Si el vector $\boldsymbol{x}$ cumple con la restricción *x* en la fórmula (12), se dice que la solución es *factible*.

2. Si se tiene el resultado máximo de $\boldsymbol{z}$, entonces se tiene que $\boldsymbol{x}$ es *óptimo*.

5.4 COMPLEJIDAD

El problema de la mochila ha sido objeto de estudio y de investigaciones debido a su complejidad; analizando desde el punto de vista de la informática este problema es interesante por las siguientes razones:

- El problema de decisión hace parte del problema de la mochila, por lo que no es posible que exista un algoritmo que genere una respuesta rápida y exacta en tiempo polinómico.

- El problema de la mochila pertenece a los problemas NP-Completos.

- Existe un tiempo pseudo-polinomial para resolverlo empleando programación dinámica.

6. IMPLEMENTACIÓN ALGORITMO GÉNETICO DE CHU-BEASLEY (AGCB)

6.1 INTRODUCCIÓN

El algoritmo genético de Chu-Beasley ha sido empleado en la solución de problemas de gran complejidad como son los pertenecientes a la clase NP-Hard, debido a su capacidad para dar soluciones óptimas. Este algoritmo al igual que los algoritmos genéticos tradicionales se basa en el proceso genético biológico en el cual se logra mantener la diversidad de los individuos que conforman una población, esto se hace mediante la ejecución de procesos como la selección, la recombinación y la mutación, en donde cada generación se reemplaza un individuo siempre que este cumpla con condiciones de optimalidad factibilidad previamente establecidas.

En el presente trabajo se realiza la implementación del algoritmo genético de Chu-Beasley para la solución de dos de los problemas de optimización más importantes, que han sido objeto de estudio e investigación a lo largo de la historia, los cuales son “El Problema del Agente Viajero (TSP)” y “El Problema de Ruteo de Vehículos”; para estos problemas no se ha encontrado un algoritmo que los resuelva de manera exacta, pero el AGCB gracias a sus características y a las variaciones que posee, en comparación con los algoritmos genéticos tradicionales, permitirá encontrar soluciones óptimas.

La implementación del algoritmo se realizó en lenguaje de programación C; en esta sección se mostrará los procesos que se llevaron a cabo en su desarrollo (selección, recombinación y mutación), así como la codificación de los datos empleados en la búsqueda de la soluciones del problema del agente viajero TSP y del problema de ruteo de vehículos VRP.

6.2 DESCRIPCIÓN DEL ALGORITMO

La implementación del algoritmos genético de Chu-Beasley se realizó en lenguaje de programación C, cada uno de los procesos llevados a cabo y las funciones implementadas se describen a continuación.

SELECCIÓN

En el proceso de selección se eligen dos cromosomas “padres” con el fin de producir un hijo que posea muy buenas características y aptitudes, las cuales permitirán que pueda ingresar a la población.

En la implementación la función *SeleccionHijo*, se encargará de realizar la selección de aquel hijo que cumpla con las condiciones de factibilidad, siendo este mejor que sus padres y evitando que sea un individuo duplicado. Dado este planteamiento se puede considerar que la selección es de tipo elitista. *(Ver código en Anexo 1).*

RECOMBINACIÓN

En la recombinación el algoritmo tiene dos individuos en los cuales se indicará que cantidad de puntos se emplearan para llevar a cabo este proceso, estos serán elegidos de manera aleatoria sobre cada cromosoma, para así obtener un individuo o descendiente que posea material genético de sus padres. *(Ver código en Anexo 2)*

MUTACIÓN

Una vez es efectuada la recombinación se realiza la mutación al descendiente, el cual sufrirá una alteración en uno de sus genes. Esta implementación emplea representación binaria, lo cual realizará cambios de 0 a 1 o viceversa. *(Ver código en Anexo 3).*

MEJORAMIENTO

El algoritmo genético de Chu-Beasley (AGCB) tiene un proceso de mejoramiento que se lleva a cabo una vez son realizados los procesos de selección, recombinación y mutación, con el fin de realizar un análisis y mejora al individuo descendiente antes de ser reemplazado. *(Ver código en Anexo 4)*

7. IMPLEMENTACIÓN DEL PROBLEMA DE LA MOCHILA

7.1 INTRODUCCIÓN

El problema de la mochila ha sido de gran interés para los investigadores, ya que este posee una estructura sencilla y características combinatorias que permiten realizar la evaluación de los algoritmos existentes en cuanto a su precisión, rapidez y robustez. Por este motivo se realizó la implementación de este en lenguaje de programación C, con el fin de realizar una calibración del algoritmo genético de Chu-Beasly y así garantizar que las soluciones generadas efectivamente sean óptimas.

En esta sección se explicará cada una de las funciones empleadas en la implementación con la finalidad de conocer su funcionamiento, con base a los planteamientos y a la formulación que anteriormente descritos.

7.2 DESCRIPCIÓN DEL ALGORITMO

- El programa tendrá un formato de archivo en el cual se ingresarán los siguientes datos:
    - Número de objetos
    - Capacidad de la mochila
    - Pesos de cada objeto
    - Costos de cada objeto

- Se emplea la representación binaria en donde 0 representa la ausencia del objeto y 1 representa la presencia.

- Se realiza el escaneo de los datos ingresados *(Ver anexo 5)*

- Una vez se ha realizado el escaneo de los datos, el programa genera las combinaciones representadas a través de cadenas de caracteres con el código binario de los números. Luego se realiza la comprobación de las condiciones que plantea el problema de la mochila, en donde la suma de los pesos de los objetos no debe superar la capacidad de la mochila. *(Ver Anexo 6)*

- Se realiza la comprobación de si el peso calculado para la combinación cumple con la condición de capacidad de la mochila. (Ver Anexo 7)

8. PRUEBAS Y RESULTADOS

8.1 RESULTADOS OBTENIDOS

ALGORITMO GENÉTICO DE CHU-BEASLY (AGCB)

Pruebas del Problema del Agente Viajero (TSP)

Las implementaciones de los algoritmos reciben los datos mediante un archivo de texto con extensión .txt, en el cual se encuentran los datos de la siguiente manera:

Número de variables = Ciudades

Volumen Máximo = Distancia Máxima

Número de puntos de recombinación

Número de puntos de mutación

Vector de costos

Vector de volumen = Vector de distancias entre ciudades

PRUEBA NÚMERO UNO

La primera prueba tiene los siguientes datos:

Tabla 3: Datos Prueba Uno

ALGORITMO DE CHU-BEASLEY		***PROBLEMA DE LA MOCHILA***
Número de puntos Recombinación 2	***Número de puntos de Mutación:*** 2	
Número de variables (ciudades): 25		***Numero de objetos (ciudades):*** 25
Volumen máximo (Distancia Máx.): 60		***Capacidad Mochila (Distancia Máx.):*** 60
Vector de costos 10 23 8 9 12 22 11 30 9 2 6 8 17 21 32 3 18 5 4 13 14 20 40 8 9		***Vector de costos*** 10 23 8 9 12 22 11 30 9 2 6 8 17 21 32 3 18 5 4 13 14 20 40 8 9
Vector de volumen (distancias) 12 10 11 15 16 2 4 6 12 20 21 34 8 31 6 5 18 22 24 3 19 17 28 3 9		***Vector de pesos (distancia)*** 12 10 11 15 16 2 4 6 12 20 21 34 8 31 6 5 18 22 24 3 19 17 28 3 9

Tabla 4: Resultados Prueba Uno

Algoritmo Genético de Chu-Beasley				***Problema de la Mochila***		**Diferencia (Chu-Beasley)-Mochila**	
Puntos de Recombinación	***Puntos de Mutación***	***Costo***	***Distancia***	***Costo***	***Distancia***	**Costo**	**Distancia**
2	2	168	56	176	59	-8	-3
3	2	168	56			-8	-3
5	2	168	56			-8	-3

PRUEBA NÚMERO DOS

Tabla 5. Datos Prueba Dos

ALGORITMO DE CHU-BEASLEY		***PROBLEMA DE LA MOCHILA***
Número de puntos recombinación: 2	***Número de puntos de mutación:*** 2	
Número de variables (ciudades): 100		***Numero de objetos (ciudades):*** 100
Volumen máximo (Distancia Máxima): 300		***Capacidad de la mochila (Distancia Máxima):*** 300
Vector de costos 1 4 2 3 8 9 1 2 4 2 1 1 3 2 9 2 6 8 1 7 2 1 3 2 3 1 8 5 4 1 3 1 4 2 3 4 9 8 9 1 2 1 2 1 0 11 1 5 1 6 2 4 6 2 2 2 1 3 4 8 3 1 6 5 1 8 2 2 4 3 9 7 18 3 9 12 34 40 9 34 8 2 21 30 5 20 24 13 18 6 4 4 18 22 25 44 2 3 12 43		***Vector de costos*** 1 4 2 3 8 9 1 2 4 2 1 1 3 2 9 2 6 8 1 7 2 1 3 2 3 1 8 5 4 1 3 1 4 2 3 4 9 8 9 1 2 1 2 1 0 11 1 5 1 6 2 4 6 2 2 2 1 3 4 8 3 1 6 5 1 8 2 2 4 3 9 7 18 3 9 12 34 40 9 34 8 2 21 30 5 20 24 13 18 6 4 4 18 22 25 44 2 3 12 43
Vector de volumen (distancias) 1 2 5 6 2 4 6 20 21 3 4 8 3 1 6 5 1 8 2 2 2 4 3 1 9 7 2 8 3 9 10 1 4 2 3 8 9 1 2 4 2 1 1 3 2 9 2 6 8 1 7 2 1 3 2 3 1 8 5 4 1 3 1 4 2 20 21 6 7 9 8 2 2 4 3 1 2 8 3 12 30 10 2 3 1 9 9 3 3 2 1 4 5 6 8 3 4 5 2		***Vector de pesos (distancia)*** 1 2 5 6 2 4 6 20 21 3 4 8 3 1 6 5 1 8 2 2 2 4 3 1 9 7 2 8 3 9 10 1 4 2 3 8 9 1 2 4 2 1 1 3 2 9 2 6 8 1 7 2 1 3 2 3 1 8 5 4 1 3 1 4 2 20 21 6 7 9 8 2 2 4 3 1 2 8 3 12 30 10 2 3 1 9 9 3 3 2 1 4 5 6 8 3 4 5 2

Tabla 6: Resultados Prueba Dos.

Algoritmo Genético de Chu-Beasley				*Problema de la Mochila*		Diferencia (Chu-Beasley) - Mochila	
Puntos de Recombinación	***Puntos de Mutación***	***Costo***	***Distancia***	***Costo***	***Distancia***	**Costo**	**Distancia**
2	2	624	297	404	243	220	54
3	2	624	297			220	54
5	3	625	297			221	54

PRUEBA NÚMERO TRES

Esta prueba se realizó empleando los datos de Westerns Sahara[19] de 29 ciudades, en donde la solución óptima fue de 27603. Se creó el 2 de Julio de 2001 y fue resuelta el 30 de Julio del mismo año. Emplearon el método Concorde para encontrar la solución, el cual es basado en programación lineal; según los datos suministrados, el algoritmo tardó 544.65 segundos. En la figura 13 se muestra la ilustración de la solución que obtuvieron.

[19] Solving TSPs. *National Traveling Salesman Problems*. http//www.math.uwaterloo.ca

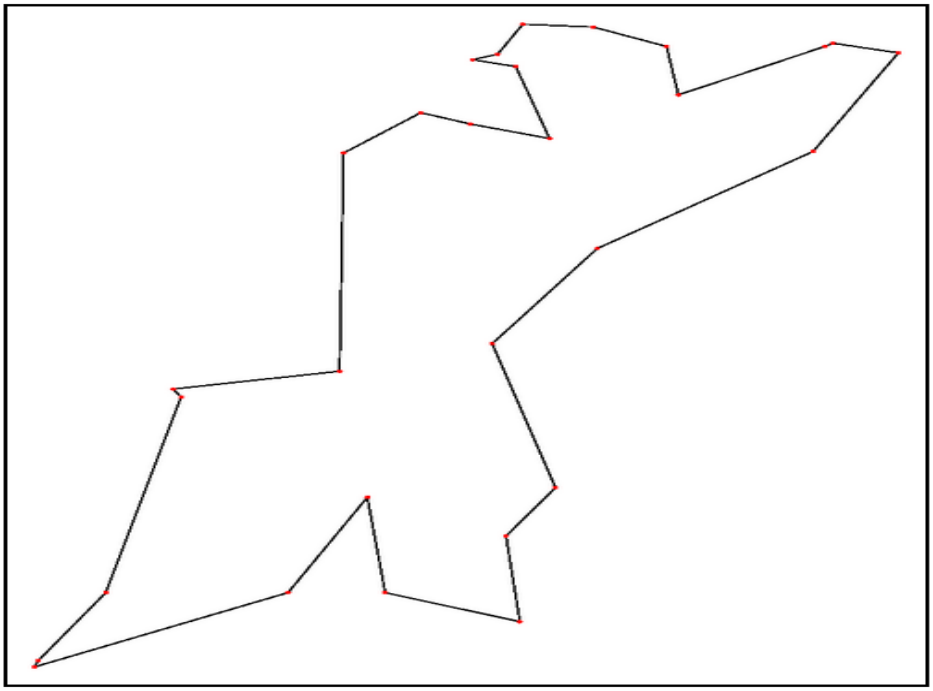

Figura 13. Solución Western Sahara

Fuente: The Traveling Salesman Problem. *National Traveling Salesman Problems. http//www.math.uwaterloo.ca*

Tabla 7: Datos Prueba Tres

ALGORITMO DE CHU-BEASLEY		***PROBLEMA DE LA MOCHILA***
Número de puntos Recombinación 2	***Número de puntos de Mutación:*** 2	
Número de variables (ciudades): 29		***Numero de objetos (ciudades):*** 29

Volumen máximo (Distancia Máxima): 27603	***Capacidad de la mochila (Distancia Máxima):*** 27603
Vector de costos 20833 20900 21300 21600 21600 21600 22183 22583 22683 23616 23700 23883 24166 13250 25149 26133 26150 26283 26433 26550 26733 27026 27096 27153 27166 27233 27266 27433 27462	***Vector de costos*** 20833 20900 21300 21600 21600 21600 22183 22583 22683 23616 23700 23883 24166 13250 25149 26133 26150 26283 26433 26550 26733 27026 27096 27153 27166 27233 27266 27433 27462
Vector de volumen (distancias) 17100 17066 13016 14150 14966 16500 13133 14300 12716 15866 15933 14533 13250 12365 14500 10550 12766 13433 13850 11683 13051 13415 13203 9833 10450 11783 10383 12400 12992	***Vector de pesos (distancia)*** 17100 17066 13016 14150 14966 16500 13133 14300 12716 15866 15933 14533 13250 12365 14500 10550 12766 13433 13850 11683 13051 13415 13203 9833 10450 11783 10383 12400 12992

Tabla 8: Resultados Prueba Tres

Algoritmo Genético de Chu-Beasley				***Problema de la Mochila***		**Diferencia (Chu-Beasley)-Mochila**	
Puntos de Recombinación	***Puntos de Mutación***	***Costo***	***Distancia***	***Costo***	***Distancia***	**Costo**	**Distancia**
2	2	54419	20216	54895	25392	-476	-5176
3	2	54419	20216			-476	-5176
5	3	54419	20216			-476	-5176

PRUEBA NÚMERO CUATRO

Tabla 9: Datos Prueba Cuatro

<table>
<tr><th colspan="2">ALGORITMO DE CHU-BEASLEY</th><th rowspan="2">PROBLEMA DE LA MOCHILA</th></tr>
<tr><th>Número de puntos Recombinación

2</th><th>Número de puntos de Mutación:

2</th></tr>
<tr><td colspan="2">Número de variables (ciudades): 100</td><td>Numero de objetos (ciudades): 100</td></tr>
<tr><td colspan="2">Volumen máximo (Distancia Máxima): 500</td><td>Capacidad de la mochila (Distancia Máxima): 500</td></tr>
<tr><td colspan="2">Vector de volumen (distancias) 12 10 11 15 15 16 2 4 6 12 20 21 34 8 31 6 5 18 22 24 3 19 17 28 3 9 10 23 8 9 12 22 11 30 9 2 6 8 17 21 32 3 18 5 4 13 14 20 40 8 9 29 8 8 23 12 28 11 7 22 9 13 5 8 8 3 21 25 16 29 19 29 1 16 10 7 15 25 7 14 22 8 9 30 7 16 28 2 19 17 29 12 29 2 19 9 16 15 5 30 23</td><td>Vector de pesos (distancia) 12 10 11 15 16 2 4 6 12 20 21 34 8 31 6 5 18 22 24 3 19 17 28 3 9 10 23 8 9 12 22 11 30 9 2 6 8 17 21 32 3 18 5 4 13 14 20 40 8 9 29 8 8 23 12 28 11 7 22 9 13 5 8 8 3 21 25 16 29 19 29 1 16 10 7 15 25 7 14 22 8 9 30 7 16 28 2 19 17 29 12 29 2 19 9 16 15 5 30 23</td></tr>
<tr><td colspan="2">Vector de costos 36 21 38 29 31 30 38 23 22 37 36 25 22 20 46 40 46 30 25 35 26 21 21 33 35 27 34 21 50 43 49 41 34 48 28 34 37 44 43 36 32 32 46 24 28 34 24 35 46 21 44 34 26 50 26 48 38 39 49 39 35 41 30 34 47 49 31 41 48 21 38 50 37 34 34 37 30 44 42 24 50 45 35 40 42 33 48 20 46 20 47 46 36 34 49 50 28 29 36 28</td><td>Vector de costos 36 21 38 29 31 30 38 23 22 37 36 25 22 20 46 40 46 30 25 35 26 21 21 33 35 27 34 21 50 43 49 41 34 48 28 34 37 44 43 36 32 32 46 24 28 34 24 35 46 21 44 34 26 50 26 48 38 39 49 39 35 41 30 34 47 49 31 41 48 21 38 50 37 34 34 37 30 44 42 24 50 45 35 40 42 33 48 20 46 20 47 46 36 34 49 50 28 29 36 28</td></tr>
</table>

Tabla 10: Resultado Prueba Cuatro

Algoritmo Genético de Chu-Beasley				***Problema de la Mochila***		**Diferencia (Chu-Beasley)-Mochila**	
Puntos de recombinación	***Puntos de Mutación***	***Costo***	***Distancia***	***Costo***	***Distancia***	**Costo**	**Distancia**
2	2	744	484	1285	470	-541	14
4	2	1217	499			-68	29

9. CONCLUSIONES

- El algoritmo genético de Chu-Beasley generó soluciones óptimas para las pruebas que se realizaron debido a sus características, las cuales permiten tener un control de la población y de cada uno de los individuos a través de los operadores genéticos.

- La implementación del problema de la mochila, permitió identificar la eficiencia del AGCB en la solución de problemas como el TSP y el VRP, dado que las soluciones generadas por ambos algoritmos fueron muy cercanas, lo cual evidencia que el manejo de la población mediante los procesos evolutivos logra resultados eficientes sin importar el tamaño o la complejidad del problema, generando soluciones en tiempos computacionales aceptables.

- Al realizar el caso de prueba de Westerns Sahara, el algoritmo genético de Chu-Beasley implementado generó una mejor solución, lo cual indica que es un algoritmo eficiente, capaz de generar soluciones óptimas, para problemas que han sido objeto de estudio y para los cuales se han implementado diferentes técnicas en busca de la mejor solución.

- La implementación del problema de la mochila y el algoritmo de Chu-Beasley realizada, puede ser aplicada en diferentes áreas, tales como la investigación de operaciones, transporte, la logística, etc., debido a la sencillez de la codificación de los datos y a la eficiencia en la generación de las soluciones.

- Las pruebas realizadas permitieron identificar que mediante el manejo de los operadores lógicos como la recombinación y la mutación, las soluciones

generadas son mejores, dado que en diferentes casos las respuestas obtenidas con puntos de recombinación de 2, 3 y 5 no generaban modificaciones significativas en las soluciones, pero una vez se realizaba un cambio en los puntos de mutación las soluciones que se obtuvieron sufrieron una variación generando resultados más óptimos.

10. BIBLIOGRAFÍA

Darwin,Charles (1859), *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life* (1.ª Edición), Londres: John Murray

Cook, J. William (2012), In Pursuit of the Traveling Salesman: *Mathematics at the Limits of Computation*, New Jersey: Princeton University Press.

G. Dantzig, R. Fulkerson y S. Johnson. Solution of large-scale traveling salesman problem*. The Rand Corporation, Santa Mónica, Californi*a. Agosto, 1954.

Aida Calviño Martínez (2011). *Cooperación en los problemas del viajante (TSP) y de rutas de vehículos (VRP): una panorámica*.

García Travieso María Victoria (2014), *Problema del viajante de comercio (TSP), Métodos exactos de resolución. Universidad de La Laguna.*

Capítulo 1: Problema del agente viajero. Recuperado de http://sedici.unlp.edu.ar

Máquina de Turing, dispositivo inventado por Alan Mathison Turing (1921-1953).

J. Perez Jiménez y Sancho Caparrini, (2003). Máquinas moleculares basadas en ADN, España.

Stockdale María Lorena (2011). El problema del viajante: un algoritmo heurístico y una aplicación. Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales. Departamento de matemáticas.

Garey R. Michael, Johnson S. David. Computer and Intractability. A guide to the Theory of NP-Completes. San Francisco: W. H Freeman and Company.

Complejidad - Problemas NP-Completos Algoritmos y Estructuras de Datos III. Http://www-2.dc.uba.ar.

Toth Paolo, Vigo Daniele (2002). The vehicle routing problem.

Dantzig G. B. y Ramser J. H. (Oct. 1959). *The truck Dispatching Problem.* (Ed. 6)

Caric Tonci y Gold Hrvoje (2008). Vehicle Ruting Problem. Croacia: In-The.

Pérez Rodríguez Jorge. Caracterización, modelado y determinación de las rutas de flota en una empresa de Rendering (p.16).

P. Toth, D. Vigo, The Vehicle Routing Problem. Monographson Discrete Mathematics and Applications. Society of Industrialand Applied Mathematics. Philadelphia. USA, 2002

Ruiz Andrés, Toro Mirledy y Salazar Harold (2007).Algoritmo genético modificado de Chu-Beasley aplicado a la identificación de errores en la estimación de estado de sistemas. (p.28)

G.B. Mathews. On the partition of numbers, Proceedings of the London Mathematical Society,

Karp Richard M. (1972). Reducibility Among Combinatorial Problems. Complexity of Computer Computations. Nueva York: Plenum. (pp. 85–103).

The Traveling Salesman Problem. National Traveling Salesman Problems. http//www.math.uwaterloo.ca

11. ANEXOS

Anexo 1. Código, implementación de Selección

```
void SeleccionHijo(int Candidatos[MAXINDIV][MAXX],  int Vol[ ], int Nvar, int B, int
NroCandidatos, int Hijos[2][MAXX])
{
 int NroHijos = 2;
 int Nh = 0;
 int R2, Elegido;
 int Duplicado;
 while (Nh < NroHijos)
  {
        srand((unsigned) time(NULL));
        Elegido = Random(NroCandidatos);
        R2 = Random(NroCandidatos);
        while (Elegido == R2)
        {
          srand((unsigned) time(NULL));
          Elegido = Random(NroCandidatos);
          R2 = Random(NroCandidatos);
        }
        if (Sinfact(Candidatos[Elegido], Vol, B, Nvar) > Sinfact(Candidatos[R2], Vol,
B, Nvar))
            Elegido = R2;
        CopiaVector(Hijos[Nh], Candidatos[Elegido], Nvar);
        if (Nh==1 )
        {
                Duplicado = IndividuoIgual(Hijos[0], Hijos[1], Nvar);
                if (Duplicado)Nh--;
                else Nh++;
        }
        else Nh++;
  }
}
```

Anexo 2. Código, implementación de Recombinación

```
void Recombinacion(int Hijos[MAXINDIV][MAXX], int HijoSeleccionado[ ], int
Nvar, int NptoRecomb)
```

```
{

          int PtoRecomb1, PtoRecomb2, Indice;
         //int NptoRecomb = (int)Nvar*0.3;
          int PtoRecomb[NptoRecomb];
          int i=0;
          int Limite = (int)Nvar/NptoRecomb;
           //Salto;
           while (i< NptoRecomb)
             {
                        srand((unsigned) time(NULL));
                        PtoRecomb[i]= Random(Limite);

                        //printf("El Pto de Recombinacion es = %d",PtoRecomb[i]
);Pausa; Salto;



                       if (i == 0)
                       {
                         Limite += (int)Nvar/NptoRecomb;
                         i++;
                       }
                       else if(PtoRecomb[i]> PtoRecomb[i-1])
                       {
                         Limite += (int)Nvar/NptoRecomb;
                         i++;
                       }
             }
             //ImprimirVector(PtoRecomb, NptoRecomb);Salto; Pausa;
             CopiaVector(HijoSeleccionado, Hijos[0], Nvar);
             Indice = 0;
             for (int j=0; j <NptoRecomb; ++ )
             {
                        for (int k= Indice; k< PtoRecomb[j]; k++)
                         HijoSeleccionado[k] = Hijos[1][k];

                Indice= PtoRecomb[j];
             }
             //printf("ya sali de Recombinación\n"); Salto; Pausa;
}
```

Anexo 3. Código, implementación de mutación

```
void Mutacion(int HijoSeleccionado[ ], int Nvar, int Nptos)
{
int PtoMuta[Nptos];
int i= 0, Diferente;

while( i < Nptos)
{
 srand((unsigned) time(NULL));
 PtoMuta[i] = Random(Nvar);
 //printf("Pto de mutacion %d", PtoMuta[i]); Salto;Pausa;
 if (i == 0)
       i++;
 else
 {
  Diferente = FALSO;
  for (int k=0; (k< i) && !Diferente;k++)
   if(PtoMuta[k] == PtoMuta[i])Diferente = VERDADERO;
  if (!Diferente)i++;
 }
}
//ImprimirVector(PtoMuta, Nptos);Salto; Pausa;
for (int j=0; j < Nptos; j++)
  HijoSeleccionado[PtoMuta[j]]= !HijoSeleccionado[PtoMuta[j]];
}
```

Anexo 4 Código, implementación Mejoramiento

```
int Mejoria(int HijoSeleccionado[], int Vol[ ], int Costo[ ],int NroCandidatos, int B, int
Nvar)
{
        int Mejorado =FALSE;
```

```
        int VolHijo = CalculaVol(Vol,HijoSeleccionado, Nvar);
        int CosHijo = CalculaFO(Costo,HijoSeleccionado,Nvar);

        if (abs(VolHijo - B) > 2)
        {
                for(int i=0; (i< Nvar) && (!Mejorado); i++)
                {
                  if (HijoSeleccionado[i])
                        {
                                if (abs (VolHijo - Vol[i]-B) < 2)
                                {
                                        HijoSeleccionado[i]=!HijoSeleccionado[i];
                                        Mejorado =VERDADERO;
                                }
                        }
                }
        }
        else
        {
                if (VolHijo == 0) return Mejorado;
                else
                {
                        for(int i=0; (i< Nvar) && (!Mejorado); i++)
                {
                  if (HijoSeleccionado[i])
                        {
                         if (abs (VolHijo - Vol[i]-B) > 2)
                                {
                                        HijoSeleccionado[i]=!HijoSeleccionado[i];
                                        Mejorado =VERDADERO;
                                }
                        }
                }
                }
        }
        return Mejorado;
}
```

Anexo 5. Escaneo de datos

```
int main(){
        FILE* a=fopen ("in.txt", "r");
        int i, peso;
        int costo,mayor_costo=0, mejor_peso=0;
        time_t inicio,fin;

        int OBJETOS=0, CAPACIDAD, MUESTRAS;

//Se escanean: cantidad de objetos y Capacidad
        fscanf (a,"%i %i", &OBJETOS, &CAPACIDAD);
        MUESTRAS= pow(2, OBJETOS);
        //printf("%i", MUESTRAS);

        int   pesos[OBJETOS];
        int   costos[OBJETOS];

 //Se escanean pesos de los objetos
        for(i=0; i<OBJETOS; i++){

                fscanf (a,"%i", &pesos[i]);
                //printf("Pesos[%i]: %i\n",i, pesos[i]);
        }


//Se escanean costos de los objetos
        for(i=0; i<OBJETOS; i++){

                fscanf (a,"%i", &costos[i]);
                //printf("Costos[%i]: %i\n",i, costos[i]);
        }
```

Anexo 6. Código, implementación de condiciones

```
for(i=0; i<MUESTRAS; i++){
                generar_vector(i, OBJETOS, vector);
                datos=calcular_peso(OBJETOS, vector, pesos, costos);
                peso = datos.peso;
                costo=datos.costo;
 printf ("%i. %s--->Peso: %i Costo: %i\n ",i, vector, datos.peso, datos.costo)
```

Anexo 7. Código, capacidad de la mochila

```
if (peso<CAPACIDAD)
        if (costo>mayor_costo){

printf ("%s---> Peso actual: %i --Costo actual: %i,MAYOR:%i\n ", vector, datos.peso,datos.costo, mayor_costo);
                mayor_costo=costo;
                mejor_peso=peso;
                strcpy(mejor, vector);
}
  }
 fin=clock();
 printf("\nLA MEJOR COMBINACION FUE: %s, con una función de costo de %i y peso %i\n", mejor, mayor_costo, mejor_peso);
```